

## AN TOÀN BẢO MẬT THÔNG TIN

-Điểm quá trình:

+1 bài kiểm tra giữa kỳ

+Chấm bài tập

+Điểm chuyên cần

-Thi cuối kỳ: 60%

+Thi viết 60 phút

+Nếu phải thi online(vấn đáp)

# *The Imitation Game*

Film received eight nominations  
at the 87th Oscar

## Buổi 1:

Bài 1:

Lập trình nhập một chuỗi ký tự bàn phím, cộng mỗi phần tử của chuỗi với 3. Hiện chuỗi mới ra màn hình.

Lập trình khôi phục lại chuỗi ban đầu

```
#include<iostream>
```

```
using namespace std;l
```

```

int main()
{
    string s;
    cout<<"Nhap chuoi: ";getline(cin, s);
    for(int i = 0; i<s.size(); i++)
    {
        s[i] = s[i] + 3;

    }
    cout<<"Chuoi ma hoa: "<<s;
    //Giai ma
    for(int i = 0; i<s.size(); i++)
    {
        s[i] = s[i] - 3;
    }
    cout<<endl<<"Chuoi giai ma: "<<s;
}

```

## Bài 2:

-Lập trình nhập một chuỗi ký tự từ bàn phím, cộng mỗi phần tử của chuỗi với K(K là một số nguyên nhập từ bàn phím). Hiện chuỗi mới ra màn hình

-Lập trình khôi phục lại chuỗi ban đầu.

```
#include<iostream>
```

```
using namespace std;
```

```

int main()
{
    string s; int K;
    cout<<"Nhap chuoi: ";getline(cin, s);
    cout<<"Nhap K = "; cin>>K;
    for(int i = 0; i<s.size(); i++)
    {
        s[i] = s[i] + K;


    }
}

```

```

        cout<<"Chuoi ma hoa: "<<s;
        //Giai ma
        for(int i = 0; i<s.size(); i++)
        {
            s[i] = s[i] - K;
        }
        cout<<endl<<"Chuoi giai ma: "<<s;
    }
}

```

 D:\AnToanBaoMat\bai1.exe

```

Nhap chuoi: Dai hoc Thuy Loi
Nhap K = 12
Chuoi ma hoa: Pmu,t{o,`tüà,X{u
Chuoi giai ma: Dai hoc Thuy Loi
-----
Process exited after 10.33 seconds with return value 0
Press any key to continue . . .

```

P ký tự bản rõ -> E hàm mã hóa -> C bản mã hóa -> D hàm giải mã

Mật mã là mã hóa nhằm mục đích giữ bí mật

Mã hóa là biến đổi thông tin từ dạng này sang dạng khác

Vành 16 là tập hợp gồm 26 số nguyên

Vành là tập hợp các số nguyên theo vòng tròn

## Buổi 2:

### Bài 1: Caesar

-Lập trình nhập một chuỗi ký tự từ bàn phím, mã hóa chuỗi bằng thuật toán CAESAR tổng quát với khóa K nhập từ bàn phím. Hiện chuỗi mới ra màn hình.

-Lập trình giải mã để khôi phục lại chuỗi ban đầu.

```

#include<iostream> //Mat ma Caesar
using namespace std;

```

```

int Kt_So(char c) //Kí tu doi sang so
{ return c-'A'; }

```

```

char So_Kt(int n) //So sang ky tu

```

```

{ return 'A' +n; }

int main()
{
    string P, C; int K;
    cout<<"Nhap chuoi: ";getline(cin,P); C=P;
    cout<<"Nhap K = "; cin>>K;
    //Mã hóa
    for(int i = 0; i<P.size(); i++)
    {
        int m; m = Kt_So(P[i]);
        m = (m + K) % 26;
        C[i] = So_Kt(m);
    }
    cout<<"Chuoi ma hoa: "<<C;
    //giai ma
    for(int i = 0; i<C.size(); i++)
    {
        int m; m = Kt_So(C[i]);
        m = (m - K + 26) % 26;
        P[i] = So_Kt(m);
    }
    cout<<endl<<"Chuoi giai ma: "<<P;
}

```

## Bài 2: Bẻ khóa Caesar

Lập trình bẻ khóa mật mã Caesar bằng phương pháp Brute-force.

-Đầu vào chương trình là chuỗi ký tự cipher text thu được từ Bài tập 1.

-Hãy xác định khóa K đã sử dụng và nội dung của plain text ban đầu.

```

#include<iostream> // Be khoa Mat ma Caesar
using namespace std;

```

```

int Kt_So(char c) //Kí tu doi sang so
{ return c-'A'; }
char So_Kt(int n) //So sang ky tu
{ return 'A'+n; }
int main()
{
    string P, C; int K;
    cout<<"Nhap chuoi: ";getline(cin,C); P=C;
    for(K = 1; K<=25; K++)
    {
        //Giai ma
        for(int i = 0; i<C.size(); i++)
        {
            int m; m = Kt_So(C[i]);
            m = (m - K + 26) % 26;
            P[i] = So_Kt(m);
        }
        cout<<K<<": "<<P <<endl;
    }
}

```

### Bài 3: a , a-1 trong $Z_{26}$

Xác định các giá trị có thể có của a trong  $Z_{26}$ , và tính a-1 tương ứng

Các giá trị của a: 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25

Các a-1 tương ứng: 1, 9, 21, 15, 3, 19, 7, 23, 11, 5, 17, 25

### Bài 4: Affine

-Lập trình nhập một chuỗi ký tự từ bàn phím, mã hóa chuỗi bằng thuật toán Affine với cặp số {a, b} nhập từ bàn phím. Hiện chuỗi mới ra màn hình.

```

#include<iostream> // Mat ma Affine
using namespace std;

```

```

int Kt_So(char c) //Kí tu doi sang so
{ return c-'A'; }
char So_Kt(int n) //So sang ky tu
{ return 'A'+n; }

```

```

int main()
{
    string P, C; int a = 0, b;
    cout<<"Nhap chuoi: ";getline(cin,P); C=P;
    while (a<1 || a>25 || a%2==0 || a==13)
    {
        cout<<"Nhap a = "; cin>>a;
    }
    cout<<"Nhap b = "; cin>>b;
    //Ma hoa
    for(int i = 0; i<P.size(); i++)
    {
        int m; m = Kt_So(P[i]);
        m = (a * m + b) % 26;
        C[i] = So_Kt(m);
    }
    cout<<"Chuoi ma hoa: "<<C;
}

```

### **Bài 5:** Giải mã Affine

Lập trình giải mã Affine để khôi phục lại chuỗi ban đầu:

-Đầu vào chương trình là cặp số {a, b} và chuỗi ký tự cipher text từ Bài tập 1.

-Đầu ra chương trình là chuỗi ký tự plain text.

```

#include<iostream> // Mat ma Affine
using namespace std;

```

```

int Kt_So(char c) //Kí tu doi sang so
{ return c-'A'; }
char So_Kt(int n) //So sang ky tu
{ return 'A'+n; }

```

```

int main()
{
    string P, C; int a = 0, b;
    cout<<"Nhap chuoi: ";getline(cin,P); C=P;
}

```

```

while (a<1 || a>25 || a%2==0 || a==13)
{
    cout<<"Nhap a = "; cin>>a;
}
cout<<"Nhap b = "; cin>>b;
//Ma hoa
for(int i = 0; i<P.size(); i++)
{
    int m; m = Kt_So(P[i]);
    m = (a * m + b) % 26;
    C[i] = So_Kt(m);
}
cout<<"Chuoi ma hoa: "<<C;
//Giai ma
int a1;
for(int i = 1; i<=25; i++)
    if((i*a)%26==1) {a1 = i;break; }
for(int i = 0; i<C.size(); i++)
{
    int m; m = Kt_So(C[i]);
    m = a1*(m-b+26)%26;
    P[i]= So_Kt(m);
}
cout<<endl<<"Chuoi giai ma: "<<P;
}

```

### Buổi 3:

#### Bài 1: Bẻ khóa Affine

Lập trình bẻ khóa mật mã Affine bằng phương pháp Brute-force:

-Đầu vào chương trình là chuỗi ký tự cipher text thu được từ Bài tập 1.

-Hãy xác định cặp số  $\{a, b\}$  đã sử dụng và nội dung của plain text ban đầu.

```
#include<iostream> // Be khoa Mat ma Affine
```

```
using namespace std;
```

```
int Kt_So(char c) //Kí tu doi sang so
```

```

{ return c-'A'; }
char So_Kt(int n) //So sang ky tu
{ return 'A'+n; }

int main()
{
    string P, C; int a = 0, b;
    cout<<"Nhap chuoi: ";getline(cin,C); P=C;
    for(a = 1; a<=25; a = a+2)
    if (a!=13)
    {
        //Giai ma
        int a1;
        for(int i = 1; i<=25; i++)
            if((i*a)%26==1) {a1 = i;break; }
        for(b = 0; b<=25; b++)
        {
            for(int i = 0; i<C.size(); i++)
            {
                int m; m = Kt_So(C[i]);
                m = a1*(m-b+26)%26;
                P[i]= So_Kt(m);
            }
            cout<<a<<" "<<b<<": "<<P <<endl;
        }
    }
}

```

## Bài 2: Mã hóa Monoalphabetic

-Lập trình nhập một chuỗi ký tự từ bàn phím, mã hóa chuỗi bằng thuật toán Monoalphabetic với khóa K nhập từ bàn phím (Khóa K là một chuỗi gồm 26 chữ cái có trật tự bất kỳ). Hiện chuỗi mới ra màn hình.

```

#include<iostream> // Mat ma MonoAiphabetic
using namespace std;

```



```

int main()
{
    string P, C, K;
    string B = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    cout<<"Nhap plaintext: ";cin>>P; C=P;
    cout<<"Nhap chuoi K = ";cin>>K;
    //Ma hoa
    for(int i = 0; i<P.size(); i++)
        for(int j = 0; j<26; j++)
            if(P[i]==B[j]) C[i]=K[j];
    cout<<"Chuoi ma hoa: "<<C;
}

```

### Bài 3: Giải mã Monoalphabetic

Lập trình giải mã Monoalphabetic để khôi phục lại chuỗi ban đầu:

-Đầu vào chương trình là khóa K và chuỗi ký tự cipher text từ Bài tập 1.

-Đầu ra chương trình là chuỗi ký tự plain text

```
#include<iostream> // Mat ma MonoAiphabetic
```

```
using namespace std;
```

```

int main()
{
    string P, C, K;
    string B = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    cout<<"Nhap plaintext: ";cin>>P; C=P;
    cout<<"Nhap chuoi K = ";cin>>K;
    //Ma hoa
    for(int i = 0; i<P.size(); i++)
        for(int j = 0; j<26; j++)
            if(P[i]==B[j]) C[i]=K[j];
    cout<<"Chuoi ma hoa: "<<C;

    //Giai ma
    for(int i = 0; i<C.size(); i++)
        for(int j = 0; j<26; j++)
            if(C[i]==K[j]) P[i]=B[j];
}

```

$$\}$$

E

- 2

S

E

B

- Chọn một từ khóa bất kỳ rồi xây dựng ma trận khóa Playfair
- Chọn một plaintext bất kỳ, áp dụng ma trận khóa trên để tạo ra ciphertext
- Thử giải mã ciphertext rồi so sánh với plaintext ban đầu

Vd của thầy

DA IH OC TH UY LO IX  
BR FB MH PD WC PM SA

BÀI 1:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

P: HOANGTHIMAI

K: GIADINH

G	I	A	D	N
H	B	C	E	F
K	L	M	O	P
Q	R	S	T	U
V	W	X	Y	Z

P = HOANGTHIMAI

P	HO	AN	GT	HI	MA	IX
C	EK	DG	DQ	BG	CX	AW

## Bài 2: Code PlayFair

- Nhập một từ khóa bất kỳ. Hãy loại bỏ các ký tự trùng lặp trong chuỗi
- Điền thêm vào chuỗi trên những ký tự còn lại trong bảng chữ cái
- Đặt chuỗi thu được vào ma trận khóa 5x5

//a)

```
#include<iostream> // Ma tran PlayFair
```

```
using namespace std;
```

```

int main()
{
    string K;
    cout<<"Nhap chuoi khoa: ";getline(cin, K);
    //Xoa trung lap

    for(int i = 0; i<K.size()-1; i++)
        for(int j = i+1; j<K.size(); j++)
            if(K[j]==K[i]) {K.erase(j, 1); j--;}
    cout<<"Chuoi ma hoa: "<<K;

```

```

}
//b)
#include<iostream> // Ma tran PlayFair
using namespace std;

```

```

int main()
{
    string K;
    string B = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    cout<<"Nhap chuoi khoa: ";getline(cin, K);
    K = K+B;
    //Xoa trung lap

    for(int i = 0; i<K.size()-1; i++)
        for(int j = i+1; j<K.size(); j++)
            if(K[j]==K[i]) {K.erase(j, 1); j--;}
    cout<<"Chuoi vua xoa: "<<K;

```

```

}

//c)
#include<iostream> // Ma tran PlayFair
using namespace std;

```

```

int main()
{
    string K;
    string B = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    cout<<"Nhap chuoi khoa: ";getline(cin, K);
    K = K+B;
    //Xoa trung lap

    for(int i = 0; i<K.size()-1; i++)
        for(int j = i+1; j<K.size(); j++)
            if(K[j]==K[i]) { K.erase(j, 1); j--;}
    cout<<"Chuoi vua xoa: "<<K <<endl;
    char A[5][5]; int t = 0;
    for(int i = 0; i<5; i++)
        for(int j = 0; j<5; j++)
            { A[i][j] = K[t]; t++; }
    //Hien ma tran khoa
    cout<<"Ma tran khoa: "<<endl;
    for(int i = 0; i<5; i++)
    {
        for(int j = 0; j<5; j++) cout<<A[i][j]<<" ";
        cout<<endl;
    }
}

```

### Buổi 5:

#### Bài 1: BT Mật mã Hill

-Với plaintext là “paymoremoney” và sử dụng với mật khóa K nói trên, hãy xác định ciphertext.

-Giải mã ciphertext thu được bằng cách nhân nó với K-1, rồi so sánh kết quả với plaintext ban đầu.

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Cách 1 :

paymoremoney

pay mor emo ney

p1p2p3

a. pay

p1 = 'p' = 15

p2 = 'a' = 0

p3 = 'y' = 24

$c1 = 17xp1 + 17xp2 + 5xp3 = 17 \times 15 + 17 \times 0 + 5 \times 24 = 375 = 11 = 'L'$

$c2 = 21xp1 + 18xp2 + 21xp3 = 21 \times 15 + 18 \times 0 + 21 \times 24 = 819 = 13 = 'N'$

$c3 = 2xp1 + 2xp2 + 19xp3 = 2 \times 15 + 2 \times 0 + 19 \times 24 = 486 = 18 = 'S'$

PAY được thay bằng LNS

Giải mã:

$p1 = c1 * 4 + c2 * 9 + c3 * 15 = 11 * 4 + 13 * 9 + 18 * 15 = 431 = 15 = 'P'$

$p2 = c1 * 15 + c2 * 17 + c3 * 6 = 11 * 15 + 13 * 17 + 18 * 6 = 494 = 0 = 'A'$

$p3 = c1 * 24 + c2 * 0 + c3 * 17 = 11 * 24 + 13 * 0 + 18 * 17 = 570 = 24 = 'Y'$

b. mor

p1 = 'm' = 12

p2 = 'o' = 14

p3 = 'r' = 17

$c1 = 17xp1 + 17xp2 + 5xp3 = 17 \times 12 + 17 \times 14 + 5 \times 17 = 527 = 7 = 'H'$

$c2 = 21xp1 + 18xp2 + 21xp3 = 21 \times 12 + 18 \times 14 + 21 \times 17 = 861 = 3 = 'D'$

$c3 = 2xp1 + 2xp2 + 19xp3 = 2 \times 12 + 2 \times 14 + 19 \times 17 = 375 = 11 = 'L'$

MOR được thay bằng HDL

Giải mã:

$$p1 = c1 * 4 + c2 * 9 + c3 * 15 = 7 * 4 + 3 * 9 + 11 * 15 = 220 = 12 = 'M'$$

$$p2 = c1 * 15 + c2 * 17 + c3 * 6 = 7 * 15 + 3 * 17 + 11 * 6 = 222 = 14 = 'O'$$

$$p3 = c1 * 24 + c2 * 0 + c3 * 17 = 7 * 24 + 3 * 0 + 11 * 17 = 355 = 17 = 'R'$$

c. emo

$$p1 = 'e' = 4$$

$$p2 = 'm' = 12$$

$$p3 = 'o' = 14$$

$$c1 = 17xp1 + 17xp2 + 5xp3 = 17x4 + 17x12 + 5x14 = 342 = 4 = 'E'$$

$$c2 = 21xp1 + 18xp2 + 21xp3 = 21x4 + 18x12 + 21x14 = 594 = 22 = 'W'$$

$$c3 = 2xp1 + 2xp2 + 19xp3 = 2x4 + 2x12 + 19x14 = 298 = 12 = 'M'$$

MOR được thay bằng EWM

Giải mã:

$$p1 = c1 * 4 + c2 * 9 + c3 * 15 = 4 * 4 + 22 * 9 + 12 * 15 = 394 = 4 = 'E'$$

$$p2 = c1 * 15 + c2 * 17 + c3 * 6 = 4 * 15 + 22 * 17 + 12 * 6 = 506 = 12 = 'M'$$

$$p3 = c1 * 24 + c2 * 0 + c3 * 17 = 4 * 24 + 22 * 0 + 12 * 17 = 300 = 14 = 'O'$$

d. ney

$$p1 = 'n' = 13$$

$$p2 = 'e' = 4$$

$$p3 = 'y' = 24$$

$$c1 = 17xp1 + 17xp2 + 5xp3 = 17x13 + 17x4 + 5x24 = 409 = 19 = 'T'$$

$$c2 = 21xp1 + 18xp2 + 21xp3 = 21x13 + 18x4 + 21x24 = 849 = 17 = 'R'$$

$$c3 = 2xp1 + 2xp2 + 19xp3 = 2x13 + 2x4 + 19x24 = 490 = 22 = 'W'$$

MOR được thay bằng TRW

Giải mã:

$$p1 = c1 * 4 + c2 * 9 + c3 * 15 = 19 * 4 + 17 * 9 + 22 * 15 = 559 = 13 = 'N'$$

$$p2 = c1 * 15 + c2 * 17 + c3 * 6 = 19 * 15 + 17 * 17 + 22 * 6 = 706 = 4 = 'E'$$

$$p3 = c1 * 24 + c2 * 0 + c3 * 17 = 19 * 24 + 17 * 0 + 22 * 17 = 830 = 24 = 'Y'$$

**Câu 2:**

Lập trình thực hiện các công việc sau:

- Khai báo chuỗi s, gán s bằng họ tên sinh viên.
- Tăng độ dài chuỗi s thành 32 kí tự bằng cách lặp lại các kí tự có trong họ tên vào cuối chuỗi.
- Sắp xếp các kí tự của chuỗi s vào một ma trận [8x4] (lần lượt theo từng cột), Đọc các phần tử của ma trận theo từng hàng để tạo thành ciphertext (đọc các hàng lẻ trước theo thứ tự 1-3-5-7, rồi đọc các hàng chẵn 0-2-4-6), hiện ciphertext ra màn hình.

Cách 2:

## 1.1 Mã hóa

P: *paymoremoney*

$$K = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

- Khóa K là ma trận 3x3  $\Rightarrow m=3$
- Chia P thành các đoạn có độ dài 3 ký tự : P= PAY MOR EMO NEY
- $P1 = [P \ A \ Y] = [15 \ 0 \ 24]$   
 $P2 = [M \ O \ R] = [12 \ 14 \ 17]$   
 $P3 = [E \ M \ O] = [4 \ 12 \ 14]$   
 $P4 = [N \ E \ Y] = [13 \ 4 \ 24]$

- $C1 = K.P1 \bmod 26$

$$= [15 \ 0 \ 24] \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \bmod 26$$

$$= [303 \ 303 \ 531] \bmod 26$$

$$= [17 \ 17 \ 11]$$

$$\rightarrow C1 = [R \ R \ L]$$



- $C2 = K.P2.\text{mod}26$



$$= [12 \ 14 \ 17] \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \text{mod}26$$

$$= [532 \ 490 \ 677] \text{mod}26$$

$$= [12 \ 22 \ 1]$$

$$\rightarrow C2 = [M \ W \ B]$$

- $C3 = K.P3.\text{mod}26$

$$= [4 \ 12 \ 14] \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \text{mod}26$$

$$= [348 \ 312 \ 538] \text{mod}26$$

$$= [10 \ 0 \ 18]$$

$$\rightarrow C3 = [K \ A \ S]$$

- $C4 = K.P4.\text{mod}26$

$$= [13 \ 4 \ 24] \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \text{mod}26$$

$$= [353 \ 341 \ 605] \text{mod}26$$

$$= [15 \ 3 \ 7]$$

$$\rightarrow C4 = [P \ D \ H]$$

$$\rightarrow C = [RRL \ MWB \ KAS \ PDH]$$

## 1.2 Giải mã

$$K^{-1} = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix}$$

- $P1 = C1 \cdot K^{-1} \text{ mod } 26$

$$= [17 \ 17 \ 11] \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \text{ mod } 26$$

$$= [587 \ 442 \ 544] \text{ mod } 26$$

$$= [15 \ 0 \ 24]$$

$$\rightarrow P1 = [P \ A \ Y]$$

- $P2 = C2 \cdot K^{-1} \text{ mod } 26$

$$= [12 \ 22 \ 1] \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \text{ mod } 26$$

$$= [402 \ 482 \ 329] \text{ mod } 26$$

$$= [12 \ 14 \ 17]$$

$$\rightarrow P2 = [M \ O \ R]$$


---

- $P3 = C3 \cdot K^{-1} \text{ mod } 26$

$$= [10 \ 0 \ 18] \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \text{ mod } 26$$

$$= [472 \ 90 \ 456] \text{ mod } 26$$

$$= [4 \ 12 \ 14]$$

$$\rightarrow P3 = [E \ M \ O]$$

- $P4 = C4 \cdot K^{-1} \text{ mod } 26$

$$= [15 \ 3 \ 7] \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \text{ mod } 26$$

$$= [273 \ 186 \ 362] \text{ mod } 26$$

$$= [13 \ 4 \ 24]$$

$$\rightarrow P4 = [N \ E \ Y]$$

$$\rightarrow P = [P \ A \ Y \ M \ O \ R \ E \ M \ O \ N \ E \ Y]$$

## Bài 2: BT Vigenere

- Chọn một plaintext và một khóa K bất kỳ
  - Áp dụng thuật toán Vigenere để xác định ciphertext
- Gợi ý: Có thể sử dụng cách tra bảng ở trang sau

plaintext: ILOVEFAMILY

ILOVEFAMILY

Viết theo dạng số là: 8 11 14 21 4 5/0 12 8 11 24

Từ khóa: MAIMAI tương ứng với:

$K = (12, 0, 8, 12, 0, 8)$

Cộng từng nhóm 6 ký tự của plaintext với K ta có:

8 11 14 21 4 5 /0 12 8 11 24

12 0 8 12 0 8 /12 0 8 12 0

20 11 22 7 4 13 / 12 12 16 23 24

Ciphertext tương ứng là:

ULWHEN/MMQXY

## Bài 3: Mã hóa Vigenere

- Lập trình nhập chuỗi ký tự plaintext từ bàn phím, mã hóa chuỗi bằng thuật toán Vigenere với khóa K (là một chuỗi ký tự) nhập từ bàn phím. Hiện chuỗi mới ra màn hình.
- Lập trình giải mã để khôi phục lại chuỗi ban đầu.

a)

```
#include<iostream> // Be khoa Mat ma Vigenere
using namespace std;
```

```
int Kt_So(char c) //Kí tự đổi sang số
```

```
{ return c-'A'; }
```

```
char So_Kt(int n) //Số sang ký tự
```

```
{ return 'A'+n; }
```

```
int main()
```

```
{
```

```
    string P, C, K;
```

```
    cout<<"Nhập chuỗi plaintext: ";cin>>P; C=P;
```

```
    cout<<"Nhập chuỗi K: "; cin>>K;
```

```

//Ma hoa
int j = 0;
for(int i = 0; i<P.size(); i++)
{
    int m; m = Kt_So(P[i]);
    int k; k = Kt_So(K[j]);
    m = (m + k)%26;
    C[i] = So_Kt(m);
    j = j+1;
    if(j==K.size()) j = 0;
}
cout<<"Chuoi ma hoa: "<<C;
}
b)
#include<iostream> // Be khoa Mat ma Vigenere
using namespace std;

int Kt_So(char c) //Kí tu doi sang so
{ return c-'A'; }
char So_Kt(int n) //So sang ky tu
{ return 'A'+n; }

int main()
{
    string P, C, K;
    cout<<"Nhap chuoi plaintext: ";cin>>P; C=P;
    cout<<"Nhap chuoi K: "; cin>>K;
    //Ma hoa
    int j = 0;
    for(int i = 0; i<P.size(); i++)
    {
        int m; m = Kt_So(P[i]);
        int k; k = Kt_So(K[j]);
        m = (m + k)%26;
        C[i] = So_Kt(m);
        j = j+1;
    }
}

```

```

        if(j==K.size()) j = 0;
    }
    cout<<"Chuoi ma hoa: "<<C;
    //Giai ma
    j = 0;
    for(int i = 0; i<C.size(); i++)
    {
        int m; m = Kt_So(C[i]);
        int k; k = Kt_So(K[j]);
        m = (m - k + 26)%26;
        P[i] = So_Kt(m);
        j = j+1;
        if(j==K.size()) j = 0;
    }
    cout<<endl<<"Chuoi giai ma: "<<P;

}

```

#### **Bài 4: BT Vernman**

-Giải sử plaintext là “HA NOI” (các ký tự trong chuỗi có thể được biểu diễn dưới dạng nhị phân bảng mã chuẩn ASCII), khóa K là một dãy nhị phân 8 bit như sau:

$$K = 10010011$$

-Hãy mã hóa chuỗi ban đầu bằng phương pháp Vernman

-Giải mã chuỗi thu được rồi so sánh với chuỗi ban đầu.

“HA NOI” = 48 41 20 4E 4F 49 = 0100 1000 0100 0001 0010 0000 0100 1110 0100  
1111 0100 1001

-Mã hóa:

Plaintext = 0100 1000 0100 0001 0010 0000 0100 1110 0100 1111 0100 1001

Khóa K = 1001 0011

Ciphertext = 1101 1011 1101 0010 1011 0011 1101 1101 1101 1100 1101 1010

-Giải mã:

Ciphertext = 1101 1011 1101 0010 1011 0011 1101 1101 1101 1100 1101 1010

Khóa K = 1001 0011 1001 0011 1001 0011 1001 0011 1001 0011 1001 0011

Plaintext = 0100 1000 0100 0001 0010 0000 0100 1110 0100 1111 0100 1001

#### **Buổi 6:**

## Bài 1: Mã hóa Vernman

- Nhập một chuỗi plaintext từ bàn phím
- Nhập một khóa K(dài 8 bit)
- Mã hóa chuỗi ban đầu bằng cách XOR các ký tự của nó với K
- Giải mã ciphertext thu được, rồi so sánh với chuỗi ban đầu.

Gợi ý: Một số phép toán thao tác với bit trong C

Phép toán	Trong C	Ví dụ
AND	&	a & b
OR		a   b
XOR	^	a ^ b
NOT	~	~a
Dịch trái	<<	a << 4
Dịch phải	>>	a >> 4

### \*CHUỖI MÃ HÓA

```
#include<iostream> // Mat ma Vernman
using namespace std;

int main()
{
    string P, C; char K;
    cout<<"Nhập plaintext: ";getline(cin, P); C=P;
    cout<<"Nhập chuỗi K = ";cin>>K;
    //Ma hoa
    for(int i = 0; i<P.size(); i++) C[i]=P[i]^K;
    cout<<"Chuỗi mã hóa: "<<C;
}
```

### \*CHUỖI GIẢI MÃ

```

#include<iostream> // Mat ma Vernman
using namespace std;

int main()
{
    string P, C; char K;
    cout<<"Nhap plaintext: ";getline(cin, P); C=P;
    cout<<"Nhap chuoi K = ";cin>>K;
    //Ma hoa
    for(int i = 0; i<P.size(); i++) C[i]=P[i]^K;
    cout<<"Chuoi ma hoa: "<<C;
    //Giai ma
    for(int i = 0; i<C.size(); i++) P[i]=C[i]^K;
    cout<<endl<<"Chuoi giai ma: "<<P;
}

```

## BÀI HOÀN THIÊN

```

#include<iostream> // Mat ma Vernman
using namespace std;

int main()
{
    string P, C, K;
    cout<<"Nhap chuoi plaintext: ";getline(cin, P); C=P;
    cout<<"Nhap chuoi K = ";getline(cin, K);
    //Ma hoa
    int j = 0;
    for(int i = 0; i<P.size(); i++)
    {
        C[i]=P[i]^K[j];
        j++;
        if(j==K.size()) j = 0;
    }
    cout<<"Chuoi ma hoa: "<<C;
    //Giai ma
    j = 0;
    for(int i = 0; i<C.size(); i++)

```

```

    {
        P[i]=C[i]^K[j];
        j++;
        if(j==K.size()) j = 0;
    }
    cout<<endl<<"Chuoi giai ma: "<<P;
}

```

## Bài 2: BT Vernman

Giả sử plaintext là “he is a nice person”

Key:	4	3	1	2	5
Plaintext:	h	e	i	s	a
	n	i	c	e	p
	e	r	s	o	n
	l	m	q	t	u
	v	w	x	y	z

Ciphertext:

icsqxseotyeirmwhnelvapnuz

ABCDEFGHIJKLMNOPQRSTUVWXYZ

## Bài 3: Mã hóa Rain-fence

-Nhập một chuỗi plaintext

-Mã hóa chuỗi bằng kỹ thuật rain-fence, hiện ciphertext ra màn hình

-Giải mã ciphertext

\*Mã hóa

```
#include<iostream> // Ky thuat Rain-fence
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    string P, C;
```

```
    cout<<"Nhap chuoi plaintext: ";getline(cin, P); C=P;
```

```
    //Ma hoa
```

```
    string h1, h2;
```

```
    for(int i = 0; i<P.size(); i++)
```

```
    {
```



```

        if(i%2==0) h1 = h1+P[i];
        else h2 = h2+P[i];
    }
    C = h1 + h2;
    cout<<"Chuoi ma hoa: "<<C;

}

*Giải mã
#include<iostream> // Ky thuat Rain-fence
using namespace std;

int main()
{
    string P, C;
    cout<<"Nhap chuoi plaintext: ";getline(cin, P); C=P;
    //Ma hoa
    string h1, h2;
    for(int i = 0; i<P.size(); i++)
    {
        if(i%2==0) h1 = h1+P[i];
        else h2 = h2+P[i];
    }
    C = h1 + h2;
    cout<<"Chuoi ma hoa: "<<C;
    //Giai ma
    P="";h1="";h2="";
    if(C.size()%2==0)
    {
        for(int i = 0; i<C.size(); i++)
        {
            if(i<C.size()/2) h1 = h1+C[i];
            else h2 = h2+ C[i];
        }
    }
    else

```

```

    {
    for(int i = 0; i<C.size(); i++)
    {
        if(i<=C.size()/2) h1 = h1+C[i];
        else h2 = h2+ C[i];
    }
}
for(int i = 0; i<C.size()/2; i++) P = P+h1[i]+h2[i];
if(C.size()%2==1) P = P+h1[h1.size()-1]; //phan tu cuoi cung cua h1
cout<<endl<<"Chuoi giai ma: "<<P;
}

```

#### Bài 4: Code đổi hàng cột.

- Nhập một chuỗi plaintext(dài không quá 25 ký tự)
- Sắp xếp các ký tự của chuỗi vào một ma trận 5x5(lần lượt theo từng hàng)
- Đọc các phần tử của ma trận theo từng cột để tạo thành ciphertext
- Giải mã ciphertext, so sánh kết quả với chuỗi ban đầu.

```

#include<iostream> // Ky thuat Doi cho hang cot
using namespace std;

```

```

int main()
{
    string P, C; char A[5][5];
    cout<<"Nhap chuoi plaintext: ";getline(cin, P);
    //Ma hoa
    int t = 0;
    for(int i = 0; i<5; i++)
        for(int j=0; j<5; j++)
        {
            A[i][j]=P[t];
            t++;
            if(t==P.size()) t=0;
        }
    for(int i = 0; i<5; i++) //Hien ma tran
    {
        for(int j = 0; j<5; j++) cout<<A[i][j]<<" ";
    }
}

```

```

        cout<<endl;
    }
    for(int i = 0; i<5; i++)
        for(int j=0; j<5; j++)
            C=C+A[j][i];

    cout<<"Chuoi ma hoa: "<<C;

}
//Giải mã
#include<iostream> // Ky thuat Doi cho hang cot
using namespace std;

int main()
{
    string P, C; char A[5][5];
    cout<<"Nhap chuoi plaintext: ";getline(cin, P);
    //Ma hoa
    int t = 0;
    for(int i = 0; i<5; i++)
        for(int j=0; j<5; j++)
        {
            A[i][j]=P[t];
            t++;
            if(t==P.size()) t=0;
        }
    for(int i = 0; i<5; i++) //Hien ma tran
    {
        for(int j = 0; j<5; j++) cout<<A[i][j]<<" ";
        cout<<endl;
    }
    for(int i = 0; i<5; i++)
        for(int j=0; j<5; j++)
            C=C+A[j][i];

    cout<<"Chuoi ma hoa: "<<C<<endl;
}

```

```

//Giai ma
t = 0;P="";
for(int i = 0; i<5; i++)
    for(int j=0; j<5; j++)
    {
        A[j][i]=C[t];
        t++;
        if(t==C.size()) t=0;
    }
for(int i = 0; i<5; i++) //Hien ma tran
{
    for(int j = 0; j<5; j++) cout<<A[i][j]<<" ";
    cout<<endl;
}
for(int i = 0; i<5; i++)
    for(int j=0; j<5; j++)
        P=P+A[i][j];

cout<<"Chuoi giai ma: "<<P;

}

```

## Buổi 7:

### Bài 1: Mật mã Feistel

Lập trình mô phỏng hoạt động của mật mã Feistel đơn giản với 2 vòng xử lý:

- Nhập một khối plaintext từ bàn phím, khối có độ dài  $m = 2w = 16$  bit.
- Nhập khóa K(đài 8 bit) từ bàn phím. Khóa  $K_i$  được sinh ra từ khóa K nhờ phép dịch trái  $K$   $i$  lần.
- Hàm F thực hiện phép cộng giữa  $R_{i-1}$  với  $K_i$ .
- Hiện khối ciphertext ra màn hình.

```

#include<iostream> // feistel
using namespace std;
char F(char R, char Ki)
{ return R+Ki; }
int main()
{

```

```

string P, C; char K[3], L[3], R[3];
cout<<"Nhap khoi plaintext: ";getline(cin, P); C=P;
cout<<"Nhap khoa K: "; cin>>K[0];
//Ma hoa
R[0]=P[0];
L[0]=P[1];
for(int i = 1; i<=2; i++)
{
    K[i]=K[0]<<i; //dich trai i lan
    R[i]=L[i-1]^F(R[i-1],K[i]);
    L[i]=R[i-1];
}
C[0]=L[2];
C[1]=R[2];
cout<<"Khoi ma hoa: "<<C;
}

```

\*Hoàn chỉnh

```

#include<iostream> // feistel
using namespace std;
char F(char R, char Ki)
{ return R+Ki; }
MHKhoi(char P0, char P1, char k)
int main()
{
    char K[3], L[3], R[3];
    string C=" ";
    //Ma hoa khoi
    R[0]=P0;
    L[0]=P1;
    K[0]=k;
    for(int i = 1; i<=2; i++)
    {
        K[i]=K[0]<<i; //dich trai i lan
        R[i]=L[i-1]^F(R[i-1],K[i]);
        L[i]=R[i-1];
    }
}

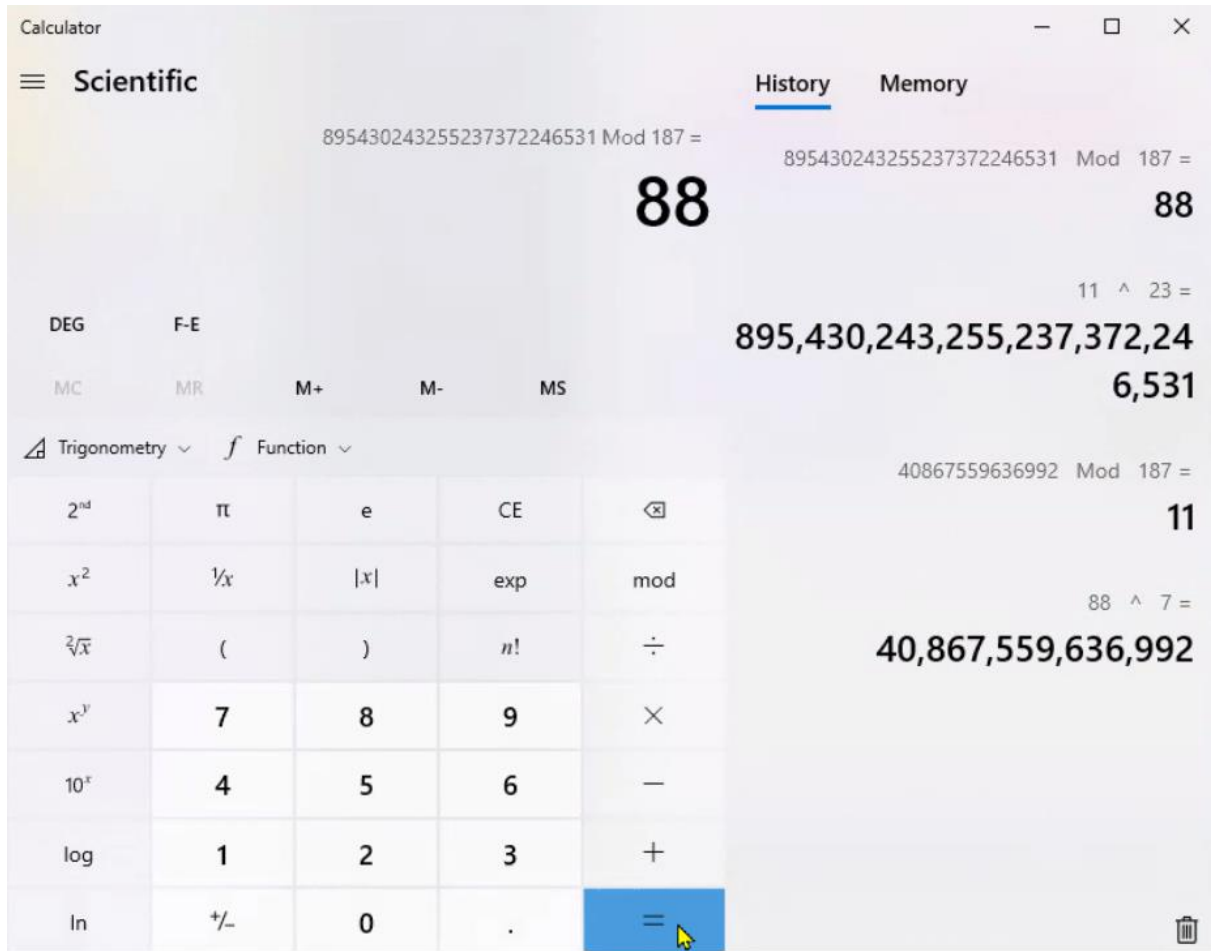
```

```

    }
    C[0]=L[2];
    C[1]=R[2];
    return C;
}
int main()
{
    string P, C; char k;
    cout<<"Nhap chuoi plaintext: ";getline(cin,P);
    cout<<"Nhap khoa K: ";cin>>k;
    if (P.size()%2==1) P=P+'X';
    for(int i = 0; i<P.size();i = i+2) C= C+MHKhoi(P[i],P[i+1],k);
    cout<<"Chuoi ma hoa: "<<C;
}

```

**Buổi 8:**



## Bài 1: RSA

- Chọn một cặp số nguyên tố p, q
- Thực hiện các bước tạo khóa, mã hóa và giải mã theo thuật toán RSA

- Chọn hai số nguyên tố,  $p = 11$  và  $q = 3$ .
- Tính  $n = p \cdot q = 11 \times 3 = 33$ .
- Tính  $\Phi(n) = (p - 1) \cdot (q - 1) = 10 \times 2 = 20$ .
- Chọn  $a$  sao cho  $a$  quan hệ nguyên tố với  $\Phi(n) = 20$  và nhỏ hơn  $\Phi(n)$ :

Ta chọn được  $a = 3$ .

- Xác định  $b$  sao cho  $a \cdot b = 1 \pmod{20}$  và  $b$  nhỏ hơn 20.
- Giá trị của  $b$  tìm được là  $b = 7$ , vì:

$$3 \times 7 = 21 \pmod{20} = 1$$

- Khoá công khai  $K1 = (a, n) = (3, 33)$
- Khoá riêng  $K2 = (b, n) = (7, 33)$

Giả sử bản rõ  $P = 15$ , khi đó quá trình mã hoá và giải mã sẽ như sau:

- Mã hoá:  $C = P^a \pmod{n} = 15^3 \pmod{33} = 9$
- Giải mã:  $P = C^b \pmod{n} = 9^7 \pmod{33} = 15$

### Ôn giữa kì

1. Lập trình
2. Mã hóa và giải mã bằng tay

SBD: 42

### Mật mã cổ điển

- 1) Kỹ thuật thay thế



- Mật mã CAESAR

## Quy tắc thay thế:

◆ Plaintext:

a b c d e f g h i j k l m n o p q r s t u v w x y z

◆ Ciphertext:

D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

- Mật mã Affine

◆ Các giá trị của a:

1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25

◆ Các  $a^{-1}$  tương ứng:

1, 9, 21, 15, 3, 19, 7, 23, 11, 5, 17, 25

- Mật mã Monoalphabetic
- Mật mã Polyalphabetic...

## 2) Kỹ thuật chuyển dịch - hoán vị

Đề 1:

Câu 1: Mật mã Vigenere

Cho chuỗi Plaintext sau : **KHONGBENHLATIENSUONGTUYETTRAN**

- Hãy mã hóa chuỗi nói trên bằng mật mã Vigenere, với khóa K là họ tên của sinh viên, m là độ dài họ tên
- Giải mã ciphertext thu được và so sánh plaintext ban đầu.

Bài làm:

**Mai:**

A. Mã hóa

-Khóa K: HOANGTHIMAI  $\Rightarrow m = 11$

-Plaintext: KHONGBENHLATIENSUONG TUYETTRAN

-Do  $m = 11$ , ta sẽ tách thành plaintext thành từng nhóm 11 ký tự:

KHONGBENHLA/TIENSUONGTU/YETTRAN

-Viết theo dạng số là:

10 7 14 13 6 1 4 13 7 11 0/19 8 4 13 18 20 14 13 6 19 20/24 4 19 19 17 0 13

-Từ khóa HOANGTHIMAI tương ứng với:

$K = (7, 14, 0, 13, 6, 19, 7, 8, 12, 0, 8)$

-Cộng từng nhóm 11 ký tự của plaintext với K ta có:

10 7 14 13 6 1 4 13 7 11 0/19 8 4 13 18 20 14 13 6 19 20/24 4 19 19 17 0 13  
 7 14 0 13 6 19 7 8 12 0 8/7 14 0 13 6 19 7 8 12 0 8/ 7 14 0 13 6 19 7  
 17 21 14 0 12 20 11 21 19 11 8/0 22 4 0 24 13 21 21 18 19 2/ 5 18 19 6 23 19 20

-Ciphertext tương ứng là:

RVOAMULVTLIAWEAYNVVSTCFSTGXTU

Giải mã

-Khóa K: HOANGTHIMAI  $\Rightarrow m = 11$

-Ciphertext: RVOAMULVTLIAWEAYNVVSTCFSTGXTU

-Do  $m = 11$ , ta sẽ tách Ciphertext thành từng nhóm 11 ký tự:

RVOAMULVTLI/AWEAYNVVSTC/FSTGXTU

-Viết theo dạng số là:

17 21 14 0 12 20 11 21 19 11 8/0 22 4 0 24 13 21 21 18 19 2/5 18 19 6 23 19 20

- Trừ từng nhóm 16 ký tự của Ciphertext với  $K$  ta có:

17 21 14 0 12 20 11 21 19 11 8/0 22 4 0 24 13 21 21 18 19 2/ 5 18 19 6 23 19 20  
7 14 0 13 6 19 7 8 12 0 8/7 14 0 13 6 19 7 8 12 0 8/ 7 14 0 13 6 19 7  
10 7 14 13 6 1 4 13 7 11 0/19 8 4 13 18 20 14 13 6 19 20/  
24 4 19 19 17 0 13

- Plaintext : KHONGBENHLATIENSUONGTUYETTRAN

### Hỏi

A. Mã hóa

- Khóa  $K$ : NGUYENTHIHONGHAI  $\Rightarrow m=16$
- Plaintext : KHONGBENHLATIENSUONGTUYETTRAN
- Do  $m=16$ , ta sẽ tách plaintext thành từng nhóm 16 ký tự:  
KHONGBENHLATIENS/UONGTUYETTRAN

- Viết theo dạng số là:

10 7 14 13 6 1 4 13 7 11 0 19 8 4 13 18 / 20 14 13 6 19 20 24 4 19 19 17 0 13

- Từ khoá NGUYENTHIHONGHAI tương ứng với:

$K = (13, 6, 20, 24, 4, 13, 19, 7, 8, 7, 14, 13, 6, 7, 0, 8)$

- Cộng từng nhóm 16 ký tự của plaintext với  $K$  ta có:

10 7 14 13 6 1 4 13 7 11 0 19 8 4 13 18 / 20 14 13 6 19 20 24 4 19 19 17 0 13  
13 6 20 24 4 13 19 7 8 7 14 13 6 7 0 8 / 13 6 20 24 4 13 19 7 8 7 14 13 6  
23 13 8 11 10 14 23 20 15 18 14 6 14 11 13 0 / 7 20 7 4 23 7 17 11 10 5 13 19

- Ciphertext tương ứng là:

XNILKOXUPSOGOLNA HUHEXHLBAFNT

B. Giải mã

- Khóa  $K$ : NGUYENTHIHONGHAI  $\Rightarrow m=16$
- Ciphertext: XNILKOXUPSOGOLNA HUHEXHLBAFNT
- Do  $m=16$ , ta sẽ tách Ciphertext thành từng nhóm 16 ký tự:

## XNILKOXUPSOGOLNA / HUHEXHRLBAFNT

- Viết theo dạng số là:

23 13 8 11 10 14 23 20 15 18 14 6 14 11 13 0/7 20 7 4 23 7 17 11 1 0 5 13 19

- Trừ từng nhóm 16 ký tự của Ciphertext với  $K$  ta có:

23 13 8 11 10 14 23 20 15 18 14 6 14 11 13 0/7 20 7 4 23 7 17 11 1 0 5 13 19

13 6 20 24 4 13 19 7 8 7 14 13 6 7 0 8/ 13 6 20 24 4 13 19 7 8 7 14 13 6

---

10 7 14 13 6 1 4 13 7 11 0 19 8 4 13 18/ 20 14 13 6 19 20 24 4 19 19 17 0 13

- Plaintext : KHONGBENHLATIENSUONGTUYETTRAN

Câu 2:

Lập trình thực hiện các công việc sau:

- Khai báo chuỗi  $s$ , gán  $s$  bằng họ tên sinh viên
- Tăng độ dài chuỗi  $s$  bằng 30 ký tự bằng cách lặp lại các ký tự trong họ tên vào cuối chuỗi
- Sắp xếp các ký tự chuỗi  $s$  vào 1 ma trận  $[3 \times 10]$  (lần lượt theo từng hàng) . Đọc các phần tử của ma trận theo từng cột để tạo thành ciphertext hiện ra màn hình.

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string s,c;
    int k[5];
    char A[5][5];
    cout<<"Nhap chuoi s: ";
    getline(cin,s);
    cout<<"Nhap khoa k: "<<endl;
    for(int i=0;i<5;i++){
        cout<<"k"<<i+1<<": ";
        cin>>k[i];
    }
    if(s.size()<25){
```

```

int dem;
dem= 25 - s.size();
int staticChuoai=0;
while (dem!=0){
s=s+s[staticChuoai];
staticChuoai++;
dem--;
}
}
int index=0;
for (int i=0;i<5;i++){
for (int j=0;j<5;j++){
A[i][j]=s[index];

cout<<A[i][j]<<" ";
index ++;
}
cout<<endl;
}
cout<<"-----"<<endl;
//ma hoa
for (int j=0;j<5;j++){
for (int i=0; i<5;i++){
c=c+A[i][k[j]-1];
}
}
cout<<endl<<"chuoi ciphertext: "<<c<<endl<<endl;

//giai ma
string giamai;
int index1=0;
for (int i=0;i<5;i++){
for (int j=0;j<5;j++){
A[j][k[i]-1]=c[index1];
index1++;
}
}

```

```

    }
}

for(int i=0;i<5;i++){
    for(int j=0;j<5;j++){
        giaima=giaima + A[i][j];
    }
    cout<<"Chuoi plaintext: "<<giaima<<endl;

    return 0;
}

```

Đề 2 :

Câu 1: Mật mã Verman

(Giải mã bằng tay)

-Giải sử plaintext là “TIEN VE HA NOI 1945” (các ký tự trong chuỗi có thể được biểu diễn dưới dạng nhị phân bảng mã chuẩn ASCII), khóa K là Họ và Tên sinh viên dưới dạng mã nhị phân. Hãy mã hóa chuỗi ban đầu bằng phương pháp Vernman

-Giải mã chuỗi thu được rồi so sánh với chuỗi ban đầu.

Khóa K = HOANG THI MAI = 48 4F 41 4E 47 20 54 48 49 20 4D 41 49

= 0100 1000 0100 1111 0100 0001 0100 1110 0100 0111 0010 0000 0101 0100 0100  
1000 0100 1001 0010 0000 0100 1101 0100 0001 0100 1001

“TIEN VE HA NOI 1945” =54 49 45 4E 20 56 45 20 48 41 20 4E 4F 49 20 31 39 34 35  
=0101 0100 0100 1001 0100 0101 0100 1110 0010 0000 0101 0110 0100 0101 0010  
0000 0100 1000 0100 0001 0010 0000 0100 1110 0100 1111 0100 1001 0011 0001 0011  
1001 0011 0100 0011 0101

-Mã hóa:

Plaintext = 0101 0100 0100 1001 0100 0101 0100 1110 0010 0000 0101 0110 0100  
0101  
0010 0000 0100 1000 0100 0001 0010 0000 0100 1110 0100 1111 0100 1001 0010 0000  
0011 0001 0011 1001 0011 0100 0011 0101

Khóa K = 0100 1000 0100 1111 0100 0001 0100 1110 0100 0111 0010 0000 0101  
0100  
0100 1000 0100 1001 0010 0000 0100 1101 0100 0001 0100 1001

Ciphertext=0001 1100 0000 0110 0000 0100 0000 0000 0110 0111 0111 0110 0001  
0001  
0110 1000 0000 0001 0110 0001 0110 1101 0000 1111 0000 0110 0000 0001 0110 1111  
0111 0000 0111 0111 0111 0011 0001 0101

-Giải mã:

Ciphertext=0001 1100 0000 0110 0000 0100 0000 0000 0110 0111 0111 0110 0001  
0001  
0110 1000 0000 0001 0110 0001 0110 1101 0000 1111 0000 0110 0000 0001 0110 1111  
0111 0000 0111 0111 0111 0011 0001 0101  
Khóa K = 0100 1000 0100 1111 0100 0001 0100 1110 0100 0111 0010 0000 0101  
0100  
0100 1000 0100 1001 0010 0000 0100 1101 0100 0001 0100 1001  
Plaintext = 0101 0100 0100 1001 0100 0101 0100 1110 0010 0000 0101 0110 0100  
0101  
0010 0000 0100 1000 0100 0001 0010 0000 0100 1110 0100 1111 0100 1001 0010 0000  
0011 0001 0011 1001 0011 0100 0011 0101

Câu 2:(Lập Trình)

-Nhập vào chuỗi S, và gán S là Họ và Tên của sinh viên

-Tăng kích thước của chuỗi S lên thành 36 ký tự

VD: ký tự của chuỗi S là ViSaoLangLe thì chuỗi S mới 36 ký tự là

ViSaoLangLeViSaoLangLeViSaoLangLeViS

-Sắp xếp các ký tự của chuỗi mới vào một ma trận 6 x 6 (lần lượt theo từng hàng)

-Đọc các phần tử của ma trận theo từng cột để tạo thành ciphertext

-Giải mã ciphertext, so sánh kết quả với chuỗi ban đầu.

Đề 3:

### Câu 1: Mật mã Affine

Mã hóa bằng tay

Cho chuỗi plaintext sau:

KHOACONGNGHETHONGTIN

-Hãy mã hóa chuỗi nói trên bằng mật mã Affine, với khóa K là cặp số  $\{a,b\}$  lấy từ 2 số cuối của mã sinh viên

- a là số hàng chục(nếu a chẵn thì cộng thêm 1)
- b là số hàng đơn vị

-Giải mã ciphertext thu được và so sánh với plaintext ban đầu.

**Mai**

-K =  $\{a,b\} = \{3, 9\}$ ,  $a^{-1} = 9$

-P = KHOACONGNGHETHONGTIN

Pi	K	H	O	A	C	O	N	G	N	G	H	E	T	H	O	N	G	T	I	N
	10	7	14	0	2	14	13	6	13	6	7	4	19	7	14	13	6	19	8	13

-Mã hóa theo công thức:

$$C_i = E(P_i, \{a,b\}) = (aP_i + b) \bmod 26 = (3P_i + 9) \bmod 26$$

=>Áp dụng công thức trên ta có:

	13	4	25	9	15	25	22	1	22	1	4	21	14	4	25	22	1	14	7	22
	N	E	Z	J	P	Z	W	B	W	B	E	V	O	E	Z	W	B	O	H	W

-Giải mã theo công thức:

$$P_i = D(C_i, \{a,b\}) = a^{-1}(C_i - b) \bmod 26 = 9(C_i - 9) \bmod 26$$

=>Áp dụng công thức trên ta có:



	N	E	Z	J	P	Z	W	B	W	B	E	V	O	E	Z	W	B	O	H	W
	10	7	14	0	2	14	13	6	13	6	7	4	19	7	14	13	6	19	8	13
	K	H	O	A	C	O	N	G	N	G	H	E	T	H	O	N	G	T	I	N

=>KL: Giải mã được kết quả trùng với Plaintext ban đầu

## Hỏi

Mật mã Affine

- $K = \{a, b\} = \{7, 1\}$ ,  $a^{-1} = 15$
- $P = \text{KHOACONGNGHETHONGTIN}$

Pi	K	H	O	A	C	O	N	G	N	G	H	E	T	H	O	N	G	T	I	N
	10	7	14	0	2	14	13	6	13	6	7	4	19	7	14	13	6	19	8	13

- **Mã hóa** theo công thức :

$$C_i = E(P_i, \{a, b\}) = (aP_i + b) \bmod 26 = (7P_i + 1) \bmod 26$$

=>Áp dụng công thức trên ta có:

C <sub>i</sub>	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>5</sub>	C <sub>6</sub>	C <sub>7</sub>	C <sub>8</sub>	C <sub>9</sub>	C <sub>10</sub>	C <sub>11</sub>	C <sub>12</sub>	C <sub>13</sub>	C <sub>14</sub>	C <sub>15</sub>	C <sub>16</sub>	C <sub>17</sub>	C <sub>18</sub>	C <sub>19</sub>	C <sub>20</sub>
	19	24	21	17	15	21	14	17	14	17	24	3	4	24	21	14	17	4	5	14
	T	Y	V	B	P	V	O	R	O	R	Y	D	E	Y	V	O	R	E	F	O

- Giải mã theo công thức:

$$P_i = D(C_i, \{a, b\}) = a^{-1}(C_i - b) \bmod 26 = 15(C_i - 1) \bmod 26$$

=>Áp dụng công thức trên ta có:

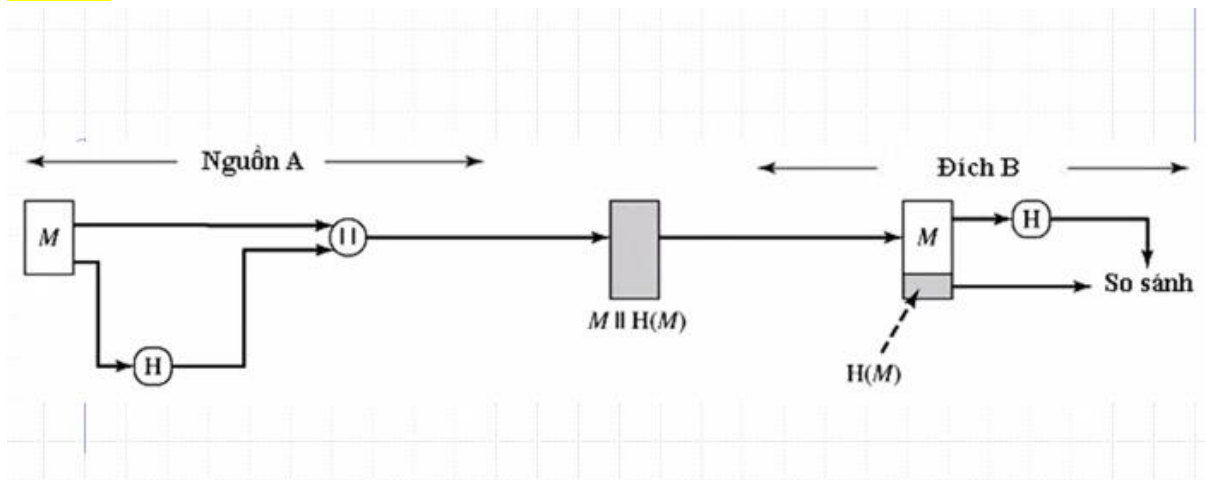
C i	T	Y	V	B	P	V	O	R	O	R	Y	D	E	Y	V	O	R	E	F	O
	1 9	2 4	2 1	1	1 5	2 1	1 4	1 7	1 4	1 7	2 4	3	4	2 4	2 1	1 4	1 7	4	5	1 4
Pi	1 0	7	1 4	0	2	1 4	1 3	6	1 3	6	7	4	1 9	7	1 4	1 3	6	1 9	8	1 3
	K	H	O	A	C	O	N	G	N	G	H	E	T	H	O	N	G	T	I	N

=>KL: Giải mã được kết quả trùng với Plaintext ban đầu

Câu 2: Lập trình thực hiện các công việc sau:

- Khai báo chuỗi khóa K, gán K bằng Họ Tên sinh viên
- Loại bỏ các ký tự trùng lặp trong K. Điền thêm vào cuối chuỗi ký tự còn lại của bảng chữ cái nhằm tạo ra chuỗi K có 26 ký tự khác nhau
- Nhập plaintext từ bàn phím mã hóa nó bằng thuật toán Monoalphabetic với khóa K nói trên.
- Giải mã ciphertext thu được và so sánh với plaintext ban đầu.

#### Buổi 9:



Biểu đồ Giúp kiểm tra thông điệp có bị lỗi trên đường truyền

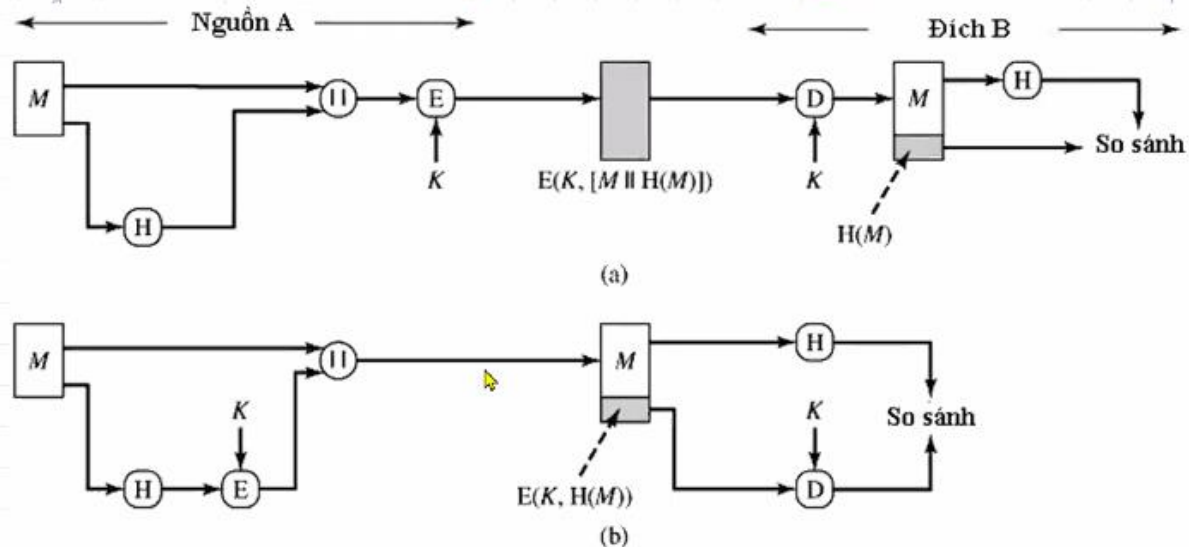
M : Dữ liệu trên đường truyền bao gồm thông điệp M được ghép với mã H(M)

Thông điệp M đc đưa qua hàm H được đưa đến bộ ghép

Bên nhận tách thành 2 phần

#### Bài 1: Hàm Hash & MM Đối xứng

## Hàm Hash và mật mã đối xứng:



Có khả năng xác thực nội dung thông điệp (thay đổi-phát hiện ra ngay) nhưng không có chức năng bảo mật nội dung

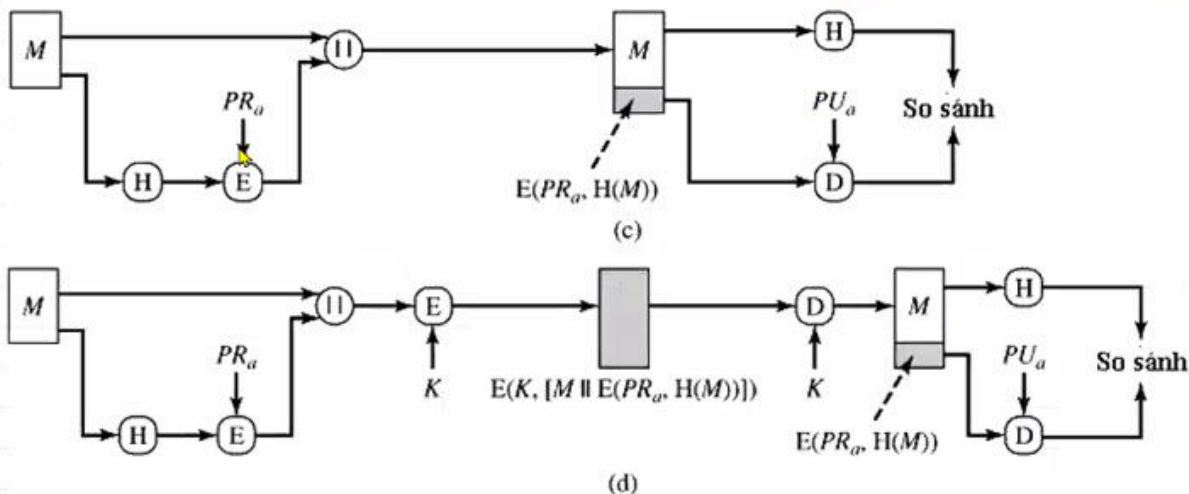
Ko cần bảo mật dùng hình a

Cần bảo mật dùng hình b

b: Tác dụng Xác thực nội dung thông điệp nhưng không có khả năng bảo mật thông điệp

- ◆ Mục đích: Đảm bảo rằng thông điệp  $M$  đến từ A và không bị thay đổi trên đường truyền (do lỗi đường truyền hoặc bị tấn công)
- ◆ Hình (b) giảm được khối lượng tính toán so với hình (a), nhưng không có khả năng giữ bí mật nội dung thông điệp như hình (a).

## Hàm Hash và mật mã khoá công khai:



$PR_a$ : (private key)- khóa riêng(giữ bí mật) của ng a  $\rightarrow$  giải mã bắt buộc phải là  $PU_a$

$PU_a$ : (public key)-khóa công khai của ng gửi a

Thông điệp  $M$  ghép với mã  $H$ . Dữ liệu đc gửi bên nhận

Nếu  $2H(M)$  khác nhau thì tức là bị lỗi

c) Dùng mật mã bất đối xứng: a dùng mã hóa riêng của mình tức là đã tạo tiền đề chữ ký số( a muốn phủ nhận cũng ko đc)

-có khả năng xác thực nội dung thông điệp,

-có khả năng xác thực nguồn gốc thông điệp m(tức là đúng từ a gửi đến)(chỉ có nguồn gốc bất đối xứng)

-Nhược điểm: Ko có khả năng bảo mật

Ví dụ:

$M \parallel E(PR_a, H(M))$

Thông điệp  $M$  được đưa qua hàm  $H$  và đưa qua hàm  $E$  để mã hóa , toàn bộ được ghép với  $M$

d) Dùng mật mã đối xứng + mật mã bất đối xứng

-Có tính năng bảo mật nội dung thông điệp(Nếu có ăn cắp thì cũng ko đọc được vì nó đã được mã hóa)

là 1 sự nâng cấp của hình c

-có khả năng xác thực nội dung thông điệp

-có khả năng xác thực nguồn gốc thông điệp m

◆ Hình (c): Đảm bảo rằng thông điệp M đến từ A và không bị thay đổi trên đường truyền, nhưng không có khả năng giữ bí mật nội dung thông điệp.

Đây chính là cơ sở của chữ kí số.

◆ Hình (d) có thêm khả năng bảo mật cho thông điệp.

**-> ứng dụng của Hash: xác thực nội dung thông điệp**

#### Bài 3: Code Hash

Nhập một chuỗi ký tự từ bàn phím, chia chuỗi thành từng khối 8 bit (ứng với 1 ký tự) rồi tính mã hash của chuỗi theo phương pháp đã nêu ở Ví dụ 1

```
#include <iostream> //Ma hash 1
using namespace std;
int main()
{
    string M; char H=0;
    cout<<"Nhap chuoi: ";getline(cin, M);

    for(int i = 0; i<M.size(); i++) H = H^M[i];

    cout<<"Ma hash: "<<H;

}
```

#### Bài 4: Tính mã Hash

Nhập một chuỗi ký tự từ bàn phím, chia chuỗi thành từng khối 64 bit (ứng với 8 ký tự) rồi tính mã hash của chuỗi theo phương pháp đã nêu ở Ví dụ 1

(Nếu độ dài chuỗi không phải là bội số của 8 thì có thể chèn thêm các ký tự quy ước.)

```
#include <iostream> //Ma hash 2
using namespace std;
int main()
{
    string M; char H[]={0,0,0,0,0,0,0,0};
    cout<<"Nhap chuoi: ";getline(cin, M);
    while (M.size()%8 !=0) M=M+'X';

    for(int i = 0; i<M.size()/8; i++)
        for(int j = 0; j<8; j++)
            H[j] = H[j]^M[i*8+j];

    /*
    {

        H[0]=H[0]^M[0];
        H[1]=H[1]^M[1];
        H[2]=H[2]^M[2];
        H[3]=H[3]^M[3];
        H[4]=H[4]^M[4];
        H[5]=H[5]^M[5];
        H[6]=H[6]^M[6];
        H[7]=H[7]^M[7];

    }
    */

    cout<<"Ma hash: ";
    for(int i = 0; i<8; i++) cout<<H[i];

}
```

Quay phải tức là dịch bit phải

**Phép dịch phải là đẩy ra khỏi dãy bit**

**Phép quay phải chính là phép dịch phải**

**Bit tận cùng là bit 1 thì chúng ta phải đưa bit ngoài cùng**

### **Ví dụ 2:**

*Giả sử thông điệp đầu vào được chia thành  $N$  khối, mỗi khối dài  $n$  bit. Để cải thiện tính ngẫu nhiên của đầu vào so với Ví dụ 1, ta sẽ thực hiện phép quay phải trước khi tiến hành XOR:*

- ◆ Giá trị hash ban đầu được đặt bằng 0
- ◆ Thực hiện  $N$  vòng lặp ( $i$  từ  $0 \rightarrow N-1$ ), mỗi vòng làm các công việc sau:
  - Quay phải một bit giá trị hash hiện tại
  - XOR giá trị hash với khối dữ liệu thứ  $i$

### **Ví dụ phép quay phải 1 dãy bit**

```
#include <iostream> //Ma hash 3
using namespace std;
int main()
{
    string M; char H=0; char n;
    cout<<"Nhập chuỗi: ";getline(cin, M);

    for(int i = 0; i<M.size(); i++)
    {
        if(H % 2 == 1) n=128; else n = 0;
        H=H<<1; H=H+n;
        H=H^M[i];
    }

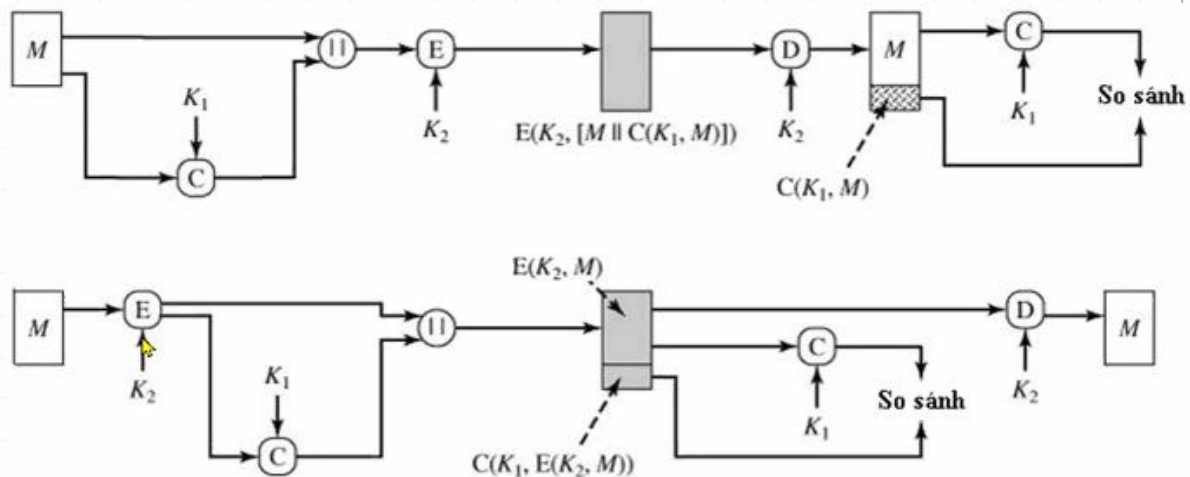
    cout<<"Ma hash: "<<H;

}
```

1	0	1	1	1	1	0	1
	1	0	1	1	1	0	

512 chia làm 8 khúc mỗi khúc 8 bit

## Một số cách dùng khác của MAC



**Giống nhau:**

- Vừa có tính xác thực nội dung thông điệp và bảo mật nội dung thông điệp

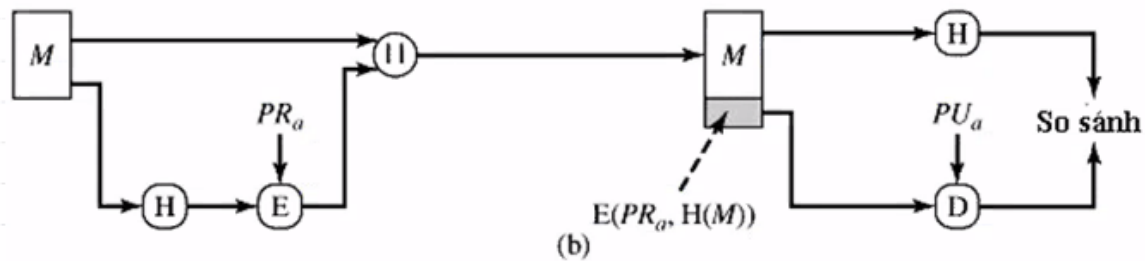
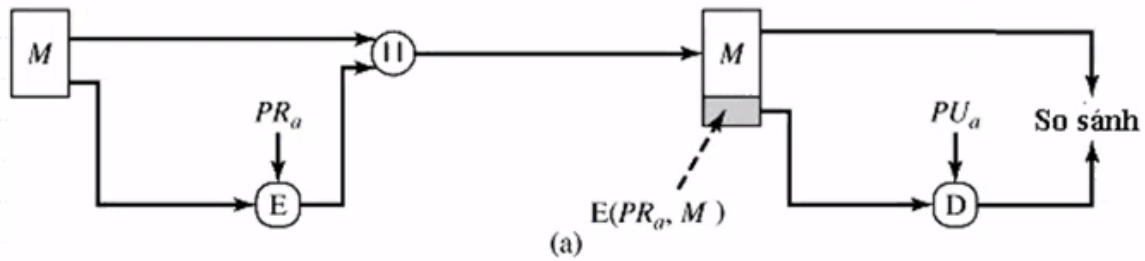
**Khác nhau:**

Sơ đồ 2 tốc độ nhanh hơn sơ đồ 1

+ Sd1 giải mã xong mới tính dc mã Mac

+ Sd2





**Giống nhau:**

**Chữ ký số:** Xác thực nội dung thông điệp và nguồn gốc thông điệp

**Khác nhau:** Sơ đồ 2 tốc độ nhanh hơn

**Sơ đồ 2:** Thông điệp M được thu ngắn lại thành 1 đoạn mã Hash ngắn

◆ Ví dụ 2:

*Mã hóa đối xứng, trọng tài không thấy nội dung thông điệp*

$$(1) X \rightarrow A: ID_X \| E(K_{xy}, M) \| E(K_{xa}, [ID_X \| H(E(K_{xy}, M))])$$

$$(2) A \rightarrow Y: E(K_{ay}, [ID_X \| E(K_{xy}, M)]) \| E(K_{xa}, [ID_X \| H(E(K_{xy}, M)) \| T])$$

◆ Ví dụ 3:

*Mã hóa khoá-công-khai, trọng tài thấy nội dung thông điệp*

$$(1) X \rightarrow A: ID_X \| M \| E(PR_x, [ID_X \| H(M)])$$

$$(2) A \rightarrow Y: E(PR_a, [ID_X \| M \| E(PR_x, [ID_X \| H(M)) \| T])$$

◆ Ví dụ 4:

*Mã hóa khoá-công-khai. Trọng tài không thấy nội dung thông điệp*

(1)  $X \rightarrow A: ID_X || E(PR_x, [ID_X || E(PU_y, E(PR_x, M))])$

(2)  $A \rightarrow Y: E(PR_a, [ID_X || E(PU_y, E(PR_x, M)) || T])$

Buổi 10

**1. Giao thức Needham-Schroeder**

Quy trình tạo ra sự kết nối có khóa phiên

## Giao thức Needham-Schroeder

1.  $A \rightarrow KDC: ID_A || ID_B || N_1$
2.  $KDC \rightarrow A: E(K_a, [K_s || ID_B || N_1 || E(K_b, [K_s || ID_A])])$
3.  $A \rightarrow B: E(K_b, [K_s || ID_A])$
4.  $B \rightarrow A: E(K_s, N_2)$
5.  $A \rightarrow B: E(K_s, f(N_2))$

## Nhận xét:

- ◆  $N_1, N_2$  là các giá trị nonce (lời gọi), có tác dụng chống tấn công nhại (đối phương chặn một thông điệp, sao chép thông tin, và sau đó gửi nhại chính thông điệp đó đến đích)
- ◆ Khi A gửi cho KDC một nonce ( $N_1$ ), theo quy ước KDC sẽ phải trả lời A bằng một giá trị tương ứng với  $N_1$ , nếu không có nghĩa là thông điệp trả lời đã bị giả mạo

## Nhược điểm:

- ◆ Chưa có cơ chế kiểm tra thời hạn sử dụng của khóa phiên  $K_s$ .
- ◆ Nếu kẻ tấn công (bằng cách nào đó) có được một khóa phiên  $K_s$  cũ, anh ta có thể dùng nó để đóng giả A nhằm lừa gạt B

## 2. Giao thức Denning

# Giao thức Denning

1.  $A \rightarrow \text{KDC}$ :  $ID_A || ID_B$
2.  $\text{KDC} \rightarrow A$ :  $E(K_a, [K_s || ID_B || T || E(K_b, [K_s || ID_A || T])])$
3.  $A \rightarrow B$ :  $E(K_b, [K_s || ID_A || T])$
4.  $B \rightarrow A$ :  $E(K_s, N_1)$
5.  $A \rightarrow B$ :  $E(K_s, f(N_1))$



## Nhận xét:

- ◆ Khóa phiên  $K_s$  được gắn kèm với nhãn thời gian  $T$
- ◆ B chỉ chấp nhận thông điệp (chứa khóa phiên  $K_s$ ) ở bước 3 nếu giá trị của nhãn thời gian  $T$  đủ gần với thời gian hiện hành của B, bao gồm độ trễ mạng và sai số cho phép
- ◆ Đồng hồ của các bên tham gia liên lạc phải được đồng bộ hóa với nhau.

## Trường hợp A, B không đồng thời trực tuyến (Chứng thực một chiều)

1.  $A \rightarrow KDC: ID_A || ID_B || N_1$
2.  $KDC \rightarrow A: E(K_a, [K_s || ID_B || N_1 || E(K_b, [K_s || ID_A])])$
3.  $A \rightarrow B: E(K_b, [K_s || ID_A]) || E(K_s, M)$

### 3. Giao thức dùng mã hóa khóa công khai

## Giao thức chứng thực dùng mã hoá khoá công khai (của Denning)

- Giữa hai bên liên lạc A và B cần có một bên thứ ba gọi là AS (Authentication Server – Server chứng thực)
- AS không sinh ra khoá phiên  $K_s$  mà sẽ đưa ra “chứng chỉ” khoá công khai
- Khoá phiên được chọn và mã hóa bởi A

```
1. A → AS:  $ID_A || ID_B$ 
2. AS → A:  $E(PR_{as}, [ID_A || PU_A || T]) || E(PR_{as}, [ID_B || PU_B || T])$ 
3. A → B:  $E(PR_{as}, [ID_A || PU_A || T]) || E(PR_{as}, [ID_B || PU_B || T]) || E(PU_B, E(PR_s, [K_s || T]))$ 
```

## Nhận xét:

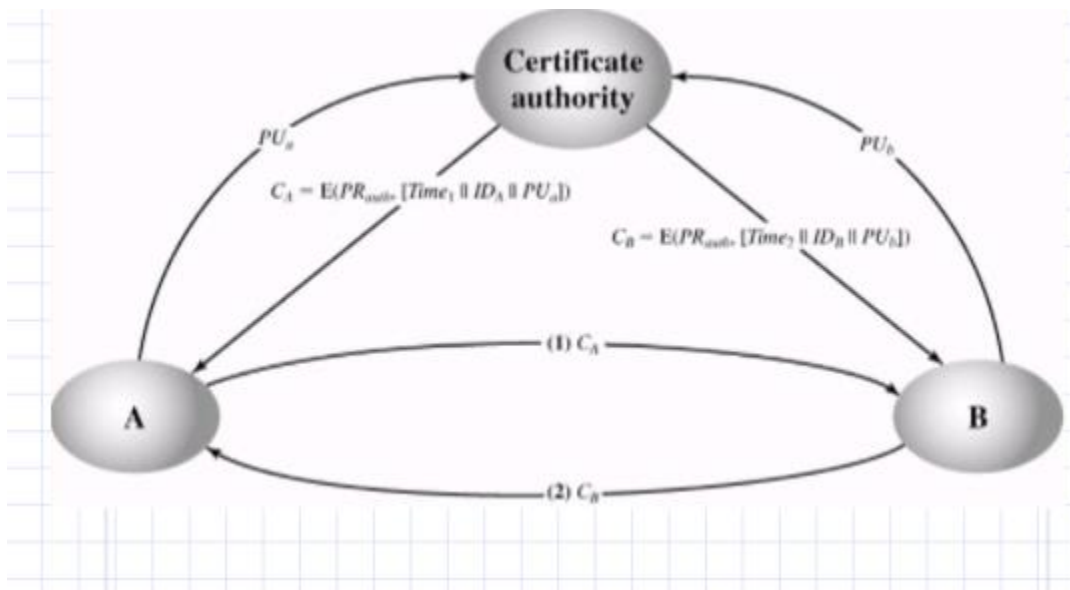
- AS sinh ra chứng chỉ khoá công khai (chứa khoá công khai của người dùng), mỗi chứng chỉ chỉ có hiệu lực trong khoảng thời gian T
- Khi A nhận được chứng chỉ do AS gửi, A sẽ biết được khoá công khai của B, và dùng nó để chuyển giao khoá phiên  $K_s$  cho B
- Khoá phiên được mã hoá 2 lần: Lần 1 bởi  $PR_{as}$  để chứng thực, lần 2 bởi  $PU_B$  để giữ bí mật

#### 4. Chứng chỉ khoá công khai

### Dùng chứng chỉ khoá công khai

- Khoá công khai không được đặt trực tiếp trong thư mục công cộng, mà được đặt trong “chứng chỉ khoá công khai”
- Người dùng A muốn công bố khoá công khai của mình thì phải đăng kí với một tổ chức đáng tin cậy có thẩm quyền cấp chứng chỉ, tổ chức này sẽ cấp cho A một chứng chỉ bao gồm khoá công khai của A ( $PU_A$ ), định danh của A ( $ID_A$ ), và chữ kí của tổ chức.

Sơ đồ





## Kí hiệu:

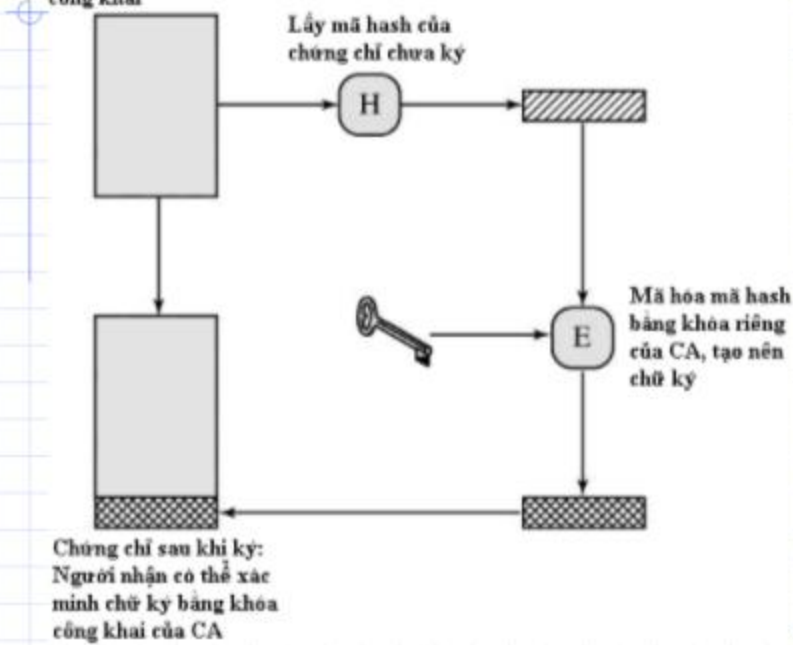
- CA: Certificate Authority (Tổ chức cấp phát chứng chỉ)
  - $C_A$ : là chứng chỉ khoá công khai của A  
$$C_A = E(PR_{auth}, [T||ID_A||PU_a])$$
  - $PR_{auth}$  là khóa riêng của tổ chức cấp phát chứng chỉ.
  - $T$  là một nhãn thời gian cho biết thời gian hiệu lực của chứng chỉ
- 
- Đối tác của A có thể dùng  $PU_{auth}$  để giải mã  $C_A$ , nhằm kiểm tra tính hợp lệ của  $C_A$  và thu được  $PU_a$ :  
$$D(PU_{auth}, C_A) = D(PU_{auth}, E(PR_{auth}, [T||ID_A||PU_a]))$$
$$= [T||ID_A||PU_a]$$
  - Sau đó đối tác có thể liên lạc với A nhờ  $PU_a$ .

### Buổi 11

Chứng chỉ X.509

# Quá trình sinh chứng chỉ X.509

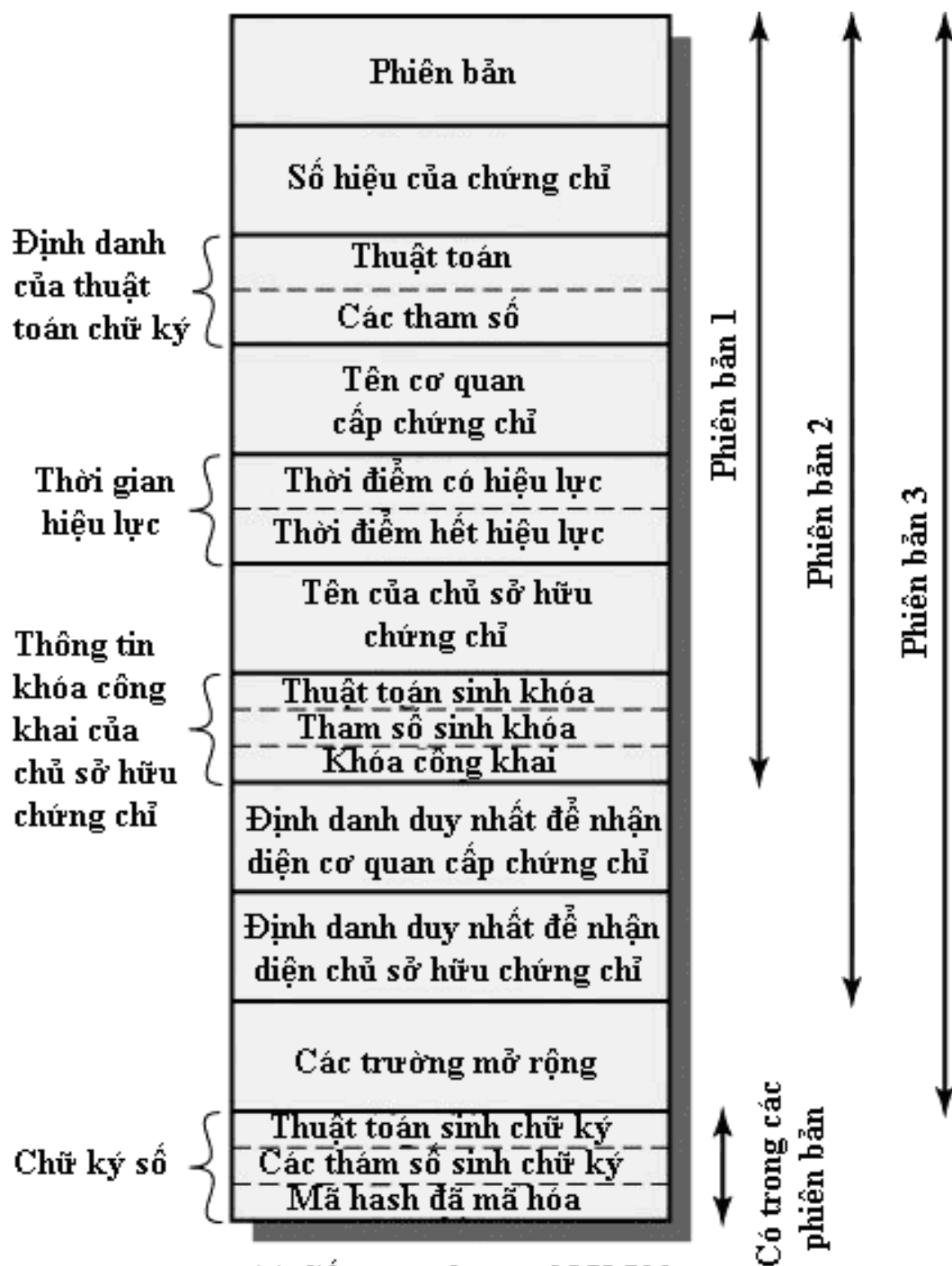
Chứng chỉ chưa ký, chứa ID người dùng và khóa công khai



ảo mật thông tin 4 - 19

$PU_a || ID_a || T || E(PR_{CA}, H([PU_a || ID_a || T]))$

Các trường

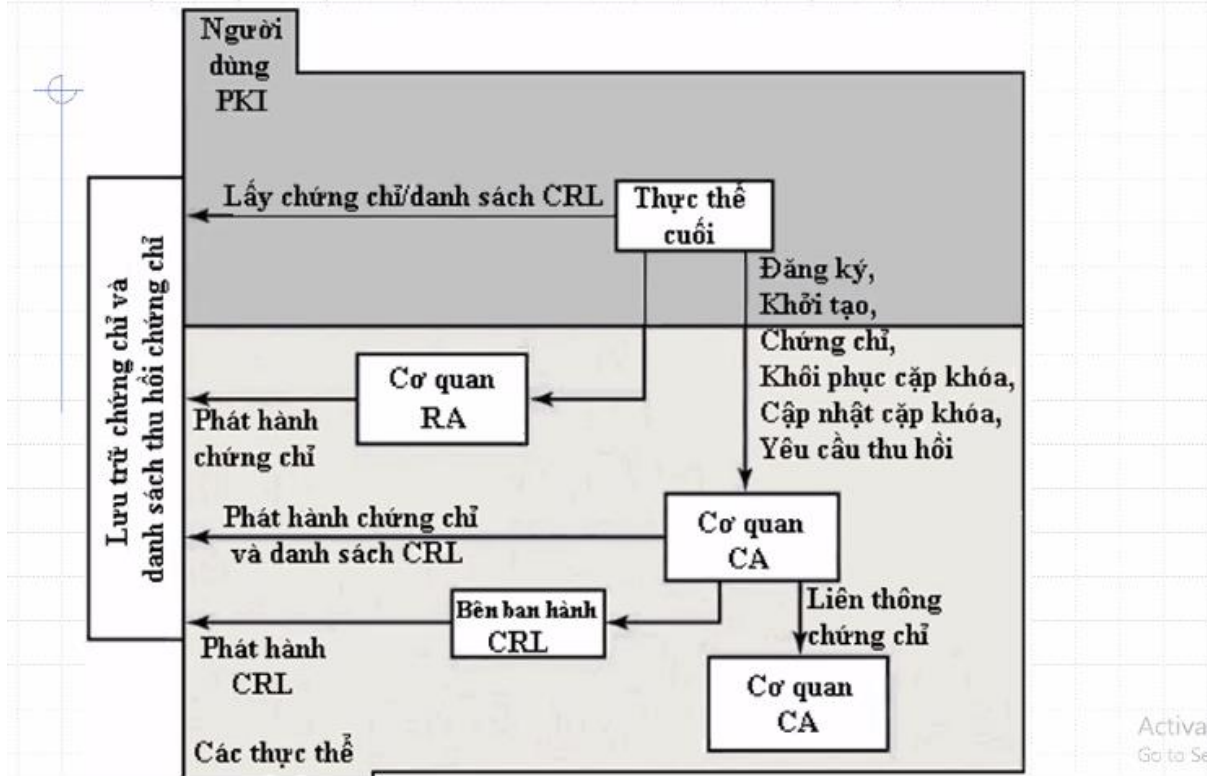


(a) Cấu trúc chứng chỉ X.509

Hạ tầng khóa công khai

Tất cả những gì cần thiết để sử dụng hệ thống mật mã khóa công khai trên thực tế

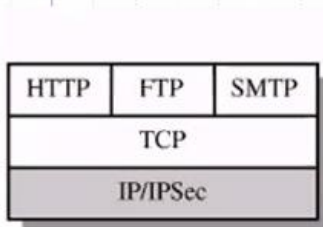
# Mô hình kiến trúc của PKIX



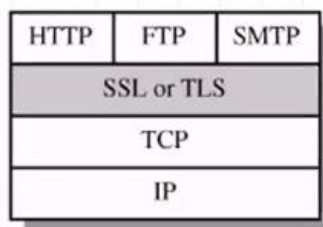
An ninh web

Giải pháp

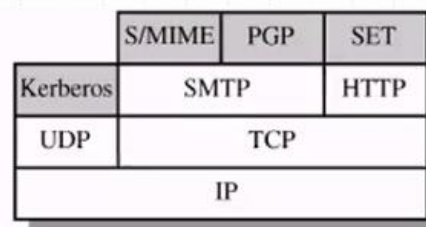
Có nhiều giải pháp an ninh cho Web ở nhiều cấp độ khác nhau:



(a) Tầng mạng



(b) Tầng chuyển vận



(c) Tầng ứng dụng

- Người dùng không được lựa chọn
- Người dùng có thể lựa chọn thích hoặc không
- Mỗi ứng dụng có 1 cách mã hóa riêng nên ít thống nhất

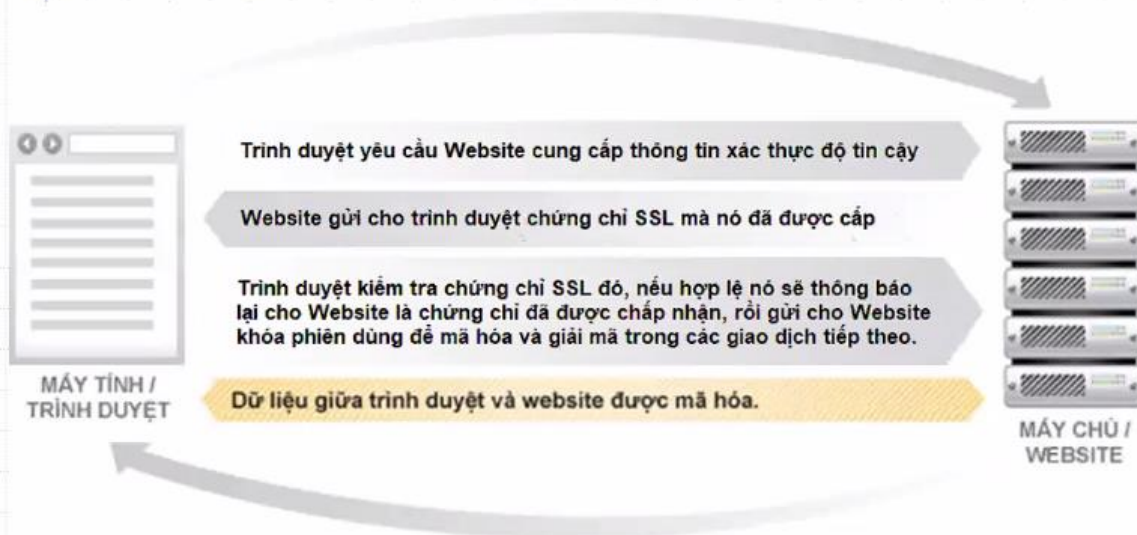
## SSL cung cấp những gì?

- ◆ Sự bảo mật: Các dữ liệu truyền tải giữa máy chủ và trình duyệt sẽ được mã hoá, đảm bảo tính riêng tư và toàn vẹn .
- ◆ Khả năng chứng thực: Mỗi chứng chỉ số SSL được tạo ra cho một Website duy nhất, khẳng định độ tin cậy của Website đó.
- ◆ Một cơ quan uy tín sẽ xác thực danh tính và độ tin cậy của Website trước khi cấp chứng chỉ SSL cho Website.

Activate W

Quy trình SSL

## Kết nối an toàn bằng SSL

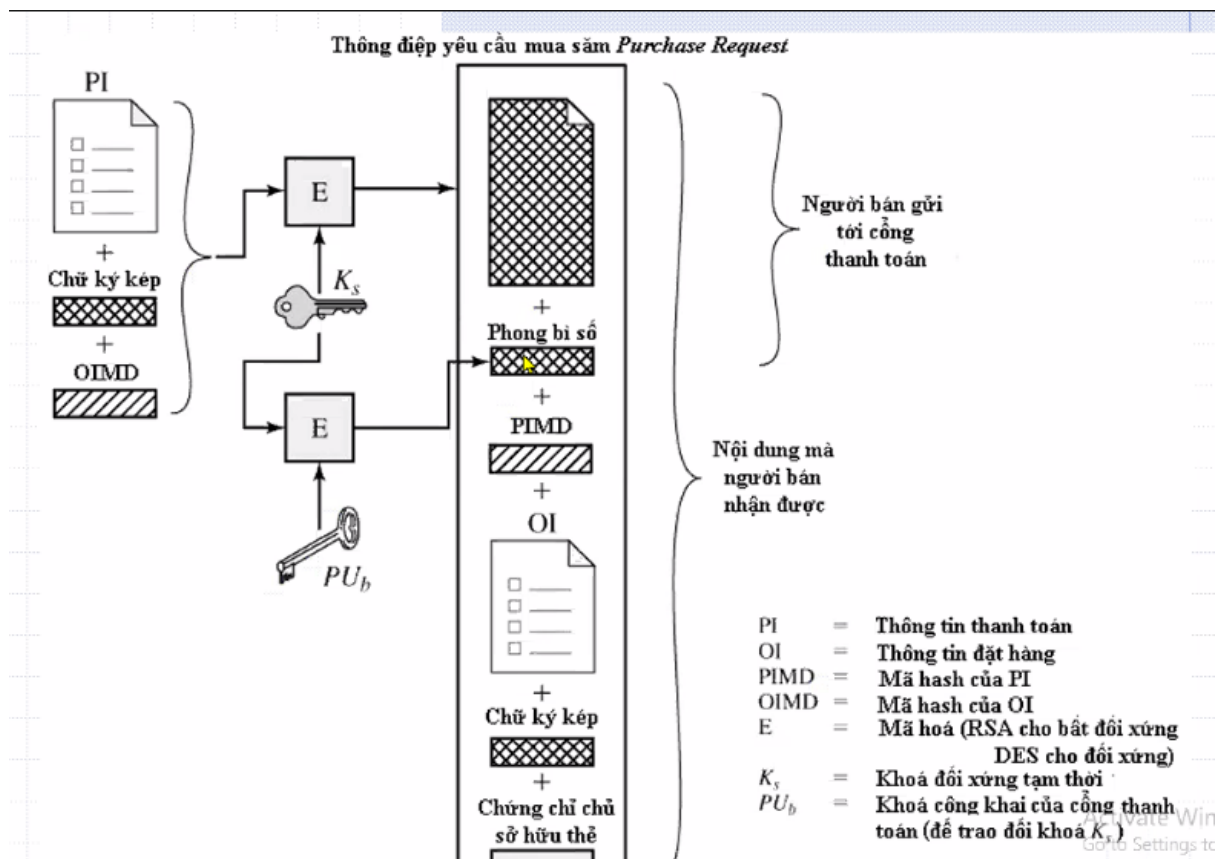
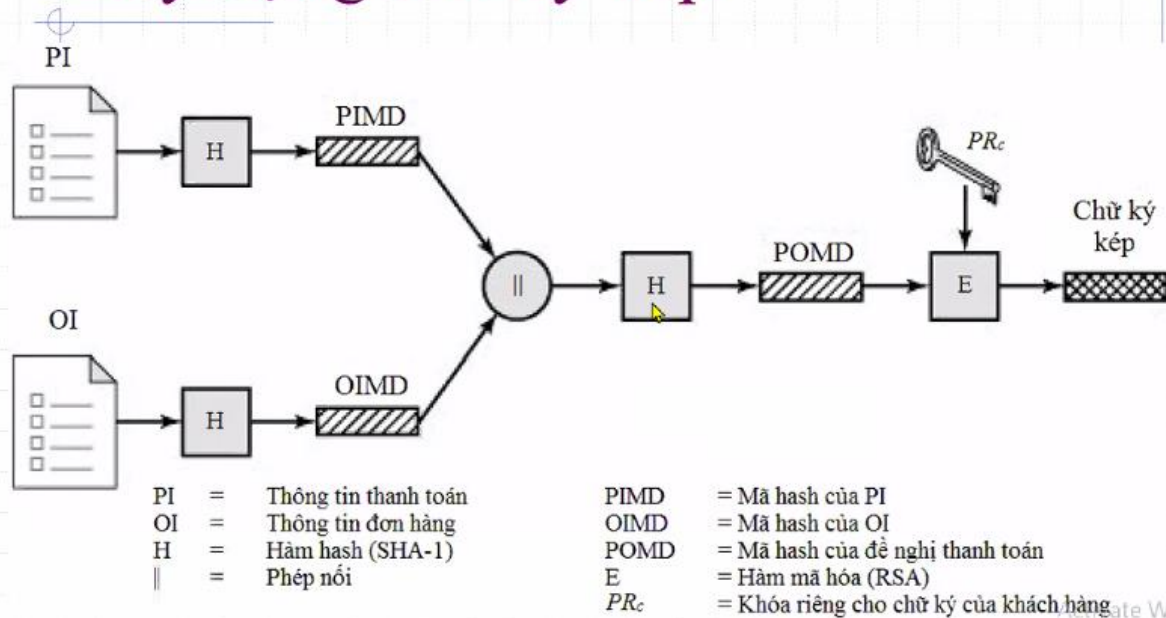


Buổi 12

An ninh giao dịch điện tử



# Xây dựng chữ ký kép



Bài tập: Vẽ sơ đồ quá trình kiểm tra chữ ký kép tại cổng thanh toán

Buổi 13

### Thi cuối kỳ

- Thi vấn đáp online
- SV đăng nhập theo cú pháp : SBD TênSV Lớp
- GV sẽ gọi SV vào theo thứ tự SBD
- Kiểm tra thẻ SV
- GV đưa câu hỏi qua màn hình share
- SV suy nghĩ(3-4 phút) và trả lời
- Nội dung: toàn bộ kiến thức đã học
- Câu hỏi 1: CT truyền tin, định dạng gói tin truyền nhận

Cho công thức biểu diễn dữ liệu trên đường truyền, hãy giải thích bên gửi và bên nhận phải làm gì, tác dụng của hệ thống đó là gì:

a:gửi, b:nhận

$M||E(P_R, H(M))$

tính mã hash của M

-bên gửi: mã hash được mã hóa bằng mã khóa riêng của a

-bên nhận tách phần mã hóa , giải mã bằng khóa công khai của ng a, giải mã xong thu được mã hash và so sánh với mã hash đính kèm. Nếu ko thay đổi thì chính xác

-tác dụng: +Chứng thực nội dung của thông điệp M có bị thay đổi

+Chứng thực thông điệp M có đúng là đến từ A ko

1 vài câu hỏi:

1. So sánh Virus và Worm
- 2) Thuật toán Vernam thực hiện ntn

### 1. Hàm Hash

-Hàm Hash là 1 công cụ quan trọng trong chứng thực

-Không thể tính ngược để thu được M từ H(M)

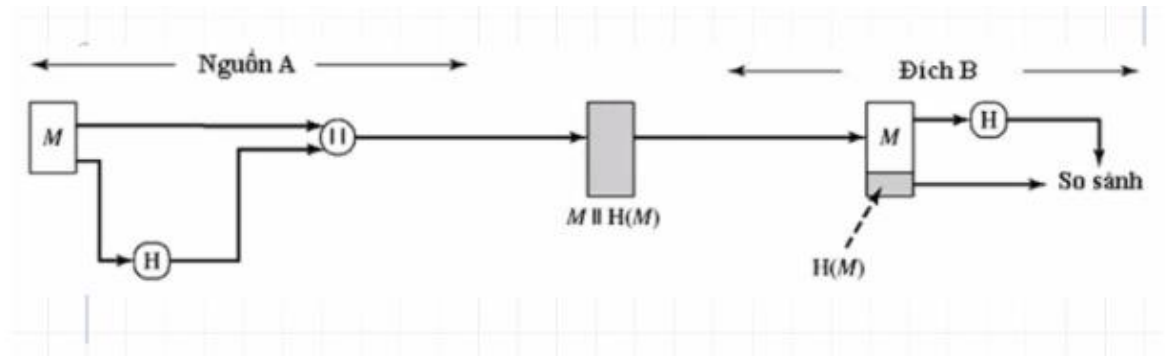
-H tác động thông điệp  $M \Rightarrow H(M)$  có kích thước , khối lượng nhỏ hơn M, có giá trị xác thực nội dung và nguồn gốc

-Hàm Hash và mật mã đối xứng mục đích: Đảm bảo rằng thông điệp M đến từ A và không bị thay đổi trên đường truyền(do lỗi đường truyền hoặc bị tấn công)

### 2. Khóa riêng, khóa công khai

- PRA ,khi mã hóa bằng khóa riêng thì lấy khóa công khai (PUa) để giải mã  
=>Tác dụng là xác thực nguồn gốc thông điệp
- PUa,khi mã hóa bằng khóa công khai thì lấy khóa riêng (PRa) để giải mã  
=>Tác dụng bảo mật

Câu 1:

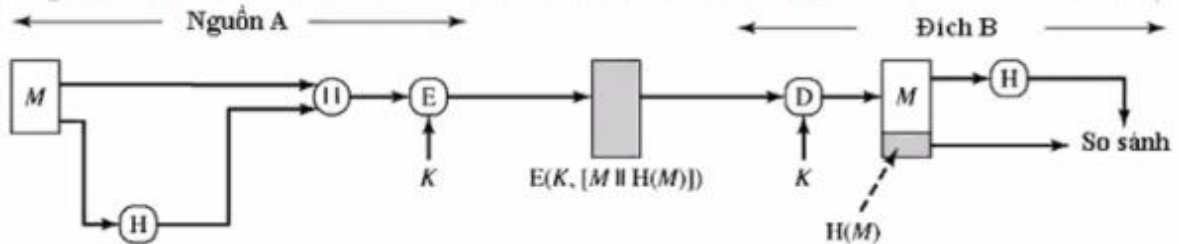


- Mục đích: Biểu đồ Giúp kiểm tra thông điệp có bị lỗi trên đường truyền
- Bên gửi:
  - Thông điệp M được đưa qua hàm Hash H thu được mã hash của M ( $H(M)$ )
  - Thông điệp M ghép với mã hash  $H(M)$  thu được  $M||H(M)$
- Bên nhận:
  - Đoạn mã nhận được thu được  $(M || H(M))$  =>Tách ra làm 2 phần
- P1: Thông điệp M đưa qua hàm hash H thu được mã  $H(M)$ (1)
- P2: Mã  $H(M)$
- Bên nhận so sánh 2 mã Hash(1) & (2) xem thông điệp nhận được có chính xác hay không?
- Tác dụng của hệ thống:
  - Xác thực nội dung thông điệp(thay đổi - phát hiện ra ngay) có chính xác hay không?(So sánh 2 mã Hash)

Câu 2: Hàm hash và mật mã đối xứng

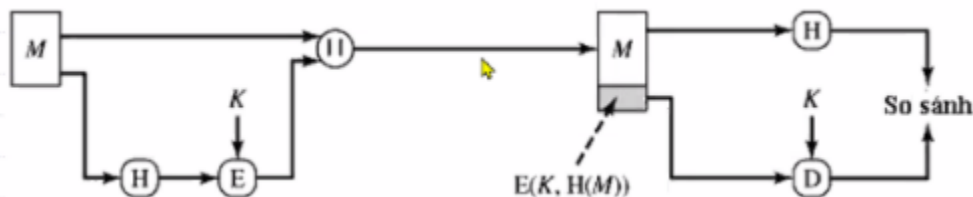


## Hàm Hash và mật mã đối xứng:



- Mục đích: Đảm bảo rằng thông điệp M đến từ A và không bị thay đổi trên đường truyền (do lỗi đường truyền hoặc bị tấn công)
  - Bên gửi:
    - Thông điệp M được đưa qua hàm Hash H thu được mã hash của M ( $H(M)$ )
    - Thông điệp M ghép với mã hash  $H(M)$  thu được  $M || H(M)$
    - Toàn bộ đoạn mã  $M || H(M)$  được mã hóa bởi hàm E với khóa K
  - Bên nhận:
    - Đoạn mã nhận được đưa qua hàm giải mã với khóa K thu được  $(M || H(M))$
- => Tách ra làm 2 phần
- P1: Thông điệp M đưa qua hàm hash H thu được mã  $H(M)$  (1)
  - P2: Mã  $H(M)$
  - Bên nhận so sánh 2 mã Hash (1) & (2) xem thông điệp nhận được có chính xác hay không?
- Tác dụng của hệ thống:
    - Xác thực nội dung thông điệp (thay đổi - phát hiện ra ngay) có chính xác hay không? (So sánh 2 mã Hash)
    - Có khả năng giữ bí mật nội dung thông điệp

Câu 3:



- Mục đích: Đảm bảo rằng thông điệp M đến từ A và không bị thay đổi trên đường truyền (do lỗi đường truyền hoặc bị tấn công)
- Bên gửi:
  - Thông điệp M được đưa qua hàm Hash H thu được mã hash của M ( $H(M)$ )

- Toàn bộ mã  $H(M)$  được mã hóa bởi hàm  $E$  với khóa  $K$
- Thông điệp  $M$  được ghép với hàm  $E(K, H(M))$

- Bên nhận:
  - Đưa thông điệp  $M$  qua hàm Hash thu được mã  $H(M)$ (1)
  - Hàm mã hóa  $E(K, H(M))$  được đưa qua hàm giải mã với khóa  $K$  thu được mã  $H(M)$  (2)
  - So sánh 2 mã Hash(1) & (2) để biết thông điệp  $M$  có chính xác hay không?
- Tác dụng của hệ thống:
  - Xác thực nội dung thông điệp(thay đổi - phát hiện ra ngay) có chính xác hay không?(So sánh 2 mã Hash)
- Nhận xét:
  - Giảm được khối lượng tính toán so với hình Câu 1
  - Không có khả năng giữ bí mật nội dung thông điệp

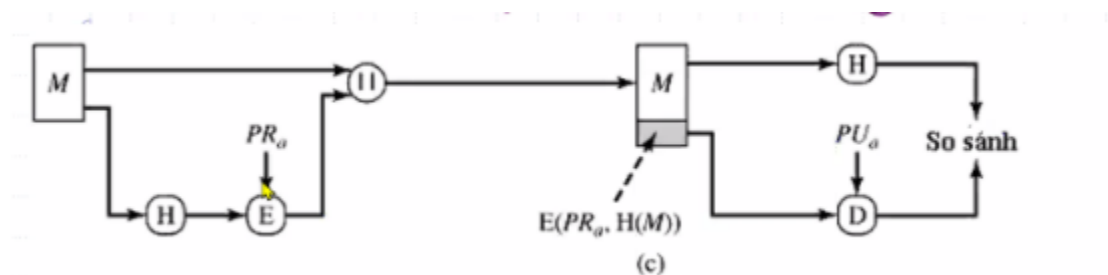
#### Câu 4.

$PR_a$ : (private key)- khóa riêng(giữ bí mật) của ng a  $\rightarrow$  giải mã bắt buộc phải là  $PU_a$

$PU_a$ : (public key)-khóa công khai của ng gửi a

Thông điệp  $M$  ghép với mã  $H$ . Dữ liệu đc gửi bên nhận

Nếu 2  $H(M)$  khác nhau thì tức là bị lỗi



- Mục đích: Đảm bảo rằng thông điệp  $M$  đến từ A và không bị thay đổi trên đường truyền
- Bên gửi:
  - Thông điệp  $M$  được đưa qua hàm Hash  $H$  thu được mã hash của  $M$  ( $H(M)$ )
  - Mã  $H(M)$  được mã hóa bằng hàm  $E$  với khóa riêng  $PR_a$
  - Thông điệp  $M$  được ghép với hàm  $E(PR_a, H(M))$

- Bên nhận:
  - Đưa thông điệp  $M$  qua hàm Hash thu được mã  $H(M)$  (1)
  - Hàm mã hóa  $E(PR_a, H(M))$  được đưa qua hàm giải mã với khóa công khai  $PU_a$  thu được mã  $H(M)$  (2)
  - So sánh 2 mã Hash (1) và (2) để biết thông điệp  $M$  có chính xác hay không?
- Tác dụng của hệ thống:
  - Xác thực nội dung thông điệp  $M$  có chính xác hay không (So sánh 2 mã Hash (1) ,(2))
  - Xác thực nguồn gốc thông điệp có phải do A gửi đến hay không ( $PR_a$ ) (chỉ có nguồn gốc bất đối xứng)
  - Tuy nhiên ko có khả năng giữ bí mật nội dung thông điệp  
Đây chính là cơ sở của chữ ký số.
- Nhận xét:

Dùng mật mã bất đối xứng: a dùng mã hóa riêng của mình tức là đã tạo tiền đề chữ ký số( a muốn phủ nhận cũng ko đc)

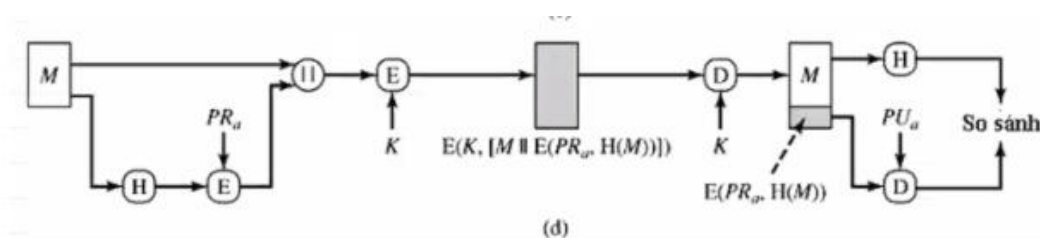
#### Câu 5.

$PR_a$ : (private key)- khóa riêng(giữ bí mật) của ng a -> giải mã bắt buộc phải là  $PU_a$

$PU_a$ : (public key)-khóa công khai của ng gửi a

Thông điệp  $M$  ghép với mã  $H$ . Dữ liệu đc gửi bên nhận

Nếu  $2H(M)$  khác nhau thì tức là bị lỗi



- Bên gửi:
  - Thông điệp  $M$  được đưa qua hàm Hash thu được mã  $H(M)$ , mã  $H(M)$  được mã hóa bằng hàm  $E$  với khóa riêng  $PR_a$  thu được  $E(PR_a, H(M))$
  - Thông điệp  $M$  ghép với hàm mã hóa  $E(PR_a, H(M))$  thu được  $(M || E(PR_a, H(M)))$

- Đoạn mã  $(M \parallel E(PRa, H(M)))$  được mã hóa bằng hàm E với khóa K.
- Bên nhận:
  - Đoạn mã nhận được đưa qua hàm giải mã với khóa K thu được  $(M \parallel E(PRa, H(M))) \Rightarrow$  Tách ra 2 phần
  - P1: Thông điệp M đưa qua hàm hash thu được mã  $H(M)$  (1)
  - P2: Hàm mã hóa  $E(PRa, H(M))$  được đưa qua hàm giải mã với khóa công khai  $PUa$  thu được  $H(M)$  (2)
  - Bên nhận so sánh 2 mã Hash (1) & (2) xem thông điệp nhận được có chính xác hay không?
- Tác dụng của hệ thống:
  - Xác thực nội dung thông điệp có chính xác hay không ? (so sánh 2 mã Hash)
  - Xác thực nguồn gốc thông điệp có phải do A gửi không (PRa)
  - Bảo mật nội dung thông điệp (Nếu có ăn cắp thì cũng ko đọc được vì nó đã được mã hóa)
- Nhận xét:
  - Là 1 sự nâng cấp của hình trên

Câu 6. Mã hóa đối xứng, trọng tài thấy nội dung thông điệp

**Ký hiệu Ý nghĩa**

$X$  = người gửi

$Y$  = người nhận

$A$  = trọng tài

$M$  = thông điệp

$T$  = timestamp, nhãn thời gian để xác định thời gian phát sinh chữ kí

$$(1) X \rightarrow A: ID_X || M || E(K_{xa}, [ID_X || H(M)])$$

$$(2) A \rightarrow Y: E(K_{ay}, [ID_X || M || E(K_{xa}, [ID_X || H(M)]) || T])$$

- Bên gửi (1):
  - X gửi A đoạn mã có : Tên người gửi X (IDx) được ghép với thông điệp M và ghép với hàm mã hóa E gồm: hàm Hash H(M) ghép với IDx sau đó đưa qua hàm mã hóa E với Khóa Kxa (X,A biết).
  - A nhận được đoạn mã từ X :
    - A tạm thời tin người gửi thông điệp là X và nhìn thấy thông điệp M.
    - Thông điệp M đưa qua hàm Hash thu được mã H(M) (1)
    - Giải mã hàm E với khóa Kxa thu được IDx và H(M) (2)
    - So sánh 2 IDx biết thông điệp do X gửi
    - So sánh 2 thông điệp Hash (1)&(2) để xác thực nội dung thông điệp không bị thay đổi.

=> A xác nhận đúng, ký và gắn nhãn thời gian T và mã hóa toàn bộ đoạn mã bằng hàm E với khóa Kay. Làm xong tất cả 3 công việc trên thì A gửi đoạn mã tới Y

- Bên nhận (2):
  - Y giải mã bằng khóa Kay (A,Y biết) từ đó biết:
    - Người gửi là A (Kay)
    - Người gửi thông điệp là X (IDx)
    - Nội dung thông điệp là M
    - Thời gian phát sinh chữ ký là T

=> X,Y phải tin tưởng tuyệt đối vào A

- Tác dụng:
  - Xác thực nội dung thông điệp
  - Xác thực nguồn gốc người gửi
- Nhận xét:
  - Cả người gửi và người nhận đều phải tin tưởng tuyệt đối vào trọng tài A
  - Nếu A không công tâm, anh ta có thể sửa đổi thông điệp, hoặc thông đồng với X để phủ nhận thông điệp đã ký, hoặc thông đồng với Y để giả mạo chữ ký của X

Câu 7. Mã hóa đối xứng, trong tài không thấy nội dung thông điệp

$$(1) X \rightarrow A: ID_X || E(K_{xy}, M) || E(K_{xa}, [ID_X || H(E(K_{xy}, M))])$$

$$(2) A \rightarrow Y: E(K_{ay}, [ID_X || E(K_{xy}, M)] || E(K_{xa}, [ID_X || H(E(K_{xy}, M)) || T])$$

- Bên gửi (1):
  - X gửi đến A đoạn mã gồm có : Tên người gửi X ghép với hàm mã hóa E mã hóa thông điệp M bằng khóa Kxy sau đó đem toàn bộ đoạn mã ghép với hàm mã hóa E gồm có thông điệp M được mã hóa bởi hàm E với khóa Kxy rồi toàn bộ hàm E(Kxy, M) đưa qua hàm Hash H thu được mã H(E(Kxy, M)) sau đó toàn bộ mã được ghép với IDx và cuối cùng toàn bộ được mã hóa bởi hàm E với khóa Kxa
  - A nhận được đoạn mã :
    - A tạm tin là người gửi thông điệp là X
    - Tiến hành giải mã hàm E với khóa Kxa đã biết thu được :IDx || H(E(Kxy,M)) (1)
    - So sánh 2 IDx xác định người gửi là X .
    - Hàm mã hóa E(Kxy,M) được đưa qua hàm Hash thu được H(E(Kxy,M)) (2)
    - So sánh 2 mã Hash 1 & 2 xác thực thông điệp không bị thay đổi trên đường truyền

=> A xác định người gửi đúng từ X và nội dung là M , A ký tên và gắn thêm nhãn thời gian T ,sau đó đem toàn bộ đoạn mã mã hóa thêm lần nữa bằng khóa Kay chuyển đoạn mã tới Y

- Bên nhận (2) :
  - Y nhận được đoạn mã từ A:
    - Tiến hành giải mã bằng khóa Kay , từ đó biết được:
  - Y biết được người gửi đoạn mã là A
    - + Người gửi thông điệp là X (IDx)
    - + Nội dung thông điệp là M (Giải mã bằng khóa Kxy)
    - + Thời gian phát sinh chữ ký là T
- Tác dụng:
  - Trọng tài A không thể đọc và giả mạo thông điệp M
  - Bảo mật hơn vì trọng tài A không đọc được nội dung của thông điệp M do được mã hóa bởi khóa Kxy
- Nhận xét:
  - Cả người gửi và người nhận đều phải tin tưởng tuyệt đối vào trọng tài A
  - A không có khả năng sửa đổi thông điệp, nhưng vẫn có thể thông đồng với X để phủ nhận thông điệp đã ký, hoặc thông đồng với Y để giả mạo chữ ký của X

Câu 8. Mã hóa khóa công khai, trọng tài thấy nội dung thông điệp

(1)  $X \rightarrow A: ID_X || M || E(PR_x, [ID_X || H(M)])$

(2)  $A \rightarrow Y: E(PR_a, [ID_X || M || E(PR_x, [ID_X || H(M)]) || T])$

- Bên gửi (1):

- X gửi A đoạn mã gồm : Tên người gửi X ghép với thông điệp M ghép với hàm mã hóa E có hàm Hash H(M) được ghép với ID<sub>x</sub> rồi đem mã hóa bằng khóa riêng của X là PR<sub>x</sub>
- A nhận được đoạn mã từ X :

- A tạm tin người gửi thông điệp là X và nhìn thấy thông điệp M

+A biết chắc rằng người gửi cho mình đúng là X(PR<sub>x</sub>)

+Đưa hàm mã hóa E qua hàm giải mã với khóa công khai PU<sub>x</sub> thu được ID<sub>x</sub> và hàm Hash H(M) (1)

+So sánh 2 ID<sub>x</sub> để biết người gửi là X

+Thông điệp M được đưa qua hàm Hash thu được H(M) (2)

+So sánh 2 mã Hash 1&2 để xác thực nội dung thông điệp M không bị thay đổi trên đường truyền.

=>A xác nhận các nội dung thông điệp nhận được từ X là đúng, tiến hành ký tên và đính kèm thời gian ký tên là T, sau đó đem toàn bộ đoạn mã trên mã hóa bằng khóa riêng của a là PR<sub>a</sub> rồi gửi đến Y.

- Bên nhận(2) :

- Y sau khi nhận được đoạn mã từ A:

+Người gửi là A (PR<sub>a</sub>)

+Giải mã bằng khóa công khai PU<sub>a</sub>:

- Sau đó biết người gửi thông điệp là X
  - Nội dung thông điệp là M
  - Thời gian phát sinh chữ ký là T

+Giải mã tiếp bằng khóa công khai PU<sub>x</sub>:

- So sánh 2 ID<sub>x</sub>
  - So sánh 2 hàm Hash H(M)

=>Y tự kiểm tra được nội dung thông điệp mà X gửi cho A

- Tác dụng:

- Xác thực nội dung thông điệp
  - Xác thực nguồn gốc thông điệp

- Nhận xét:

A biết nội dung thông điệp, nhưng không thể sửa đổi thông điệp hay làm giả chữ ký

Câu 9. Mã hóa khóa công khai, trong tài không thấy nội dung thông điệp

$$\begin{aligned}(1) X \rightarrow A: ID_X \| E(PR_x, [ID_X \| E(PU_y, E(PR_x, M))]) \\ (2) A \rightarrow Y: E(PR_a, [ID_X \| E(PU_y, E(PR_x, M)) \| T])\end{aligned}$$

- Bên gửi (1):
  - X gửi A đoạn mã gồm: Tên người gửi X ghép với hàm mã hóa E có thông điệp M được mã hóa bởi khóa riêng PR<sub>x</sub> sau đó được mã hóa bằng khóa công khai PU<sub>y</sub> rồi đem đoạn mã ghép với ID<sub>x</sub> sau đó đem toàn bộ đoạn mã mã hóa bởi khóa riêng PR<sub>x</sub>.
  - A nhận được đoạn mã từ X:
    - +Biết người gửi là X(ID<sub>x</sub>), khi nhìn thấy PR<sub>x</sub> thì biết được nguồn gốc thông điệp là từ X
    - +Giải mã bằng khóa công khai PU<sub>x</sub> thu được:
      - ID<sub>x</sub>, so sánh với ID<sub>x</sub> bên ngoài => tin rằng người gửi là X
      - Hàm mã hóa khóa công khai PU<sub>y</sub> => Không thể giải mã được vì không có khóa riêng của Y

=>A xác nhận được người gửi là X, sau đó ký tên và đính kèm thời gian ký tên là T rồi mã hóa toàn bộ đoạn mã bằng khóa riêng PR<sub>a</sub>.

- Bên nhận(2):
  - Y nhận được đoạn mã từ A :
    - Giải mã bằng khóa công khai PU<sub>a</sub>(Người gửi cho mình là A) thu được:
      - +ID<sub>x</sub>=>Người gửi thông điệp là X
      - +Thời gian ký tên là T
      - +Giải mã bằng khóa riêng PR<sub>y</sub> sau đó giải mã bằng khóa công khai PU<sub>x</sub> thu được thông điệp M.
  - Tác dụng:
    - Xác thực nội dung
    - Xác thực nguồn gốc
    - Bảo mật
  - Nhận xét:



A không biết nội dung thông điệp, không thể sửa đổi thông điệp hay làm giả chữ ký

Câu 10 Ưu nhược điểm của các kỹ thuật mã hóa cổ điển

-Kỹ thuật thay thế: Thuật toán mã hoá sẽ thay thế mỗi kí tự trong bản rõ bằng một kí tự khác

-Phân biệt Caesar, Affine, Monoalphabetic, Ponoalphabetic

Tên mật mã	Caesar	Affine	Monoalphabetic	Ponoalphabetic
Tổng quát	-Mỗi kí tự trong bảng chữ cái được thay thế bởi một kí tự khác cùng bảng, cách sau nó ba vị trí -	Kí tự $P$ ban đầu được thay thế bởi kí tự $C$	Mỗi kí tự trong bảng chữ cái được thay thế bởi một kí tự bất kì khác cùng bảng -Mật mã Monoalphabetic chỉ sử dụng một bảng mã (mỗi kí tự plain text được thay thế bởi một kí tự cố định), nên không giấu được tần suất xuất hiện các kí tự	sử dụng nhiều bảng mã khác nhau (mỗi kí tự plain text có thể được thay thế bởi nhiều kí tự khác nhau, dựa trên các khoá thay thế khác nhau)
Ưu điểm	Mật mã Caesar đơn giản, dễ thực hiện	Mật mã Affine có độ phức tạp lớn hơn mật mã Caesar tổng quát, số lượng khoá cũng nhiều hơn	Mật mã Monoalphabetic có số lượng khoá rất lớn, khó bẻ khoá bằng phương pháp Brute - force	
Nhược điểm	Độ an toàn không cao, dễ bị bẻ khoá bởi	Độ an toàn chưa cao, dễ bị phá bởi tấn	-Tuy nhiên vẫn có thể bẻ khoá mật mã này dựa trên các thống kê về các	

	tấn công Brute-force do số lượng khoá quá ít (chỉ có 25 khoá)	công Brute-force do số lượng khoá chưa nhiều (chỉ có 312 khoá)	đặc điểm tự nhiên của ngôn ngữ -Mật mã Monoalphabetic dễ bẻ vì không che giấu được tần xuất suất hiện của các kí tự trong văn bản Để khắc phục có thể áp dụng mã hoá đa kí tự, hoặc sử dụng nhiều bảng mã thay thế (Polyalphabetic)...	
--	---	--	--	--

-Kỹ thuật chuyển dịch- hoán vị: Các kí tự của plaintext sẽ được hoán đổi vị trí cho nhau để tạo thành ciphertext

Kỹ thuật thay thế	Kỹ thuật hoán vị
có tác dụng làm <i>xáo trộn</i> thông tin thống kê của plaintext, làm phức tạp hóa mối quan hệ thống kê giữa ciphertext và mật khoá, nhằm ngăn cản nỗ lực tìm khoá.	có tác dụng làm <i>khuếch tán</i> thông tin thống kê của plaintext, pha loãng các cấu trúc thống kê của plaintext ra một phạm vi rộng hơn, làm phức tạp hóa mối quan hệ thống kê giữa ciphertext và plaintext.

+Có thể lặp lại phép thay thế và hoán vị nhiều lần để tăng độ phức tạp, nâng cao tính bảo mật

-Kỹ thuật rain-fence

Tên	Kỹ thuật rain-fence
-----	---------------------

Tổng quát	Plaintext được viết dịch xuống tuần tự theo các đường chéo rồi đọc trình tự theo các hàng.
Ưu	
Nhược	<p>Mật mã hoán vị thuần túy rất dễ nhận ra bởi nó giữ nguyên tần suất xuất hiện ký tự đơn (và làm thay đổi tần suất của các cặp, các bộ ký tự của plaintext)</p> <p>Để tăng độ phức tạp, người ta có thể tiến hành đổi chỗ nhiều lần, hoặc kết hợp với các thuật toán mã hoá khác.</p>

-Mã hóa đa ký tự

-Phân biệt Mật mã Playfair, Mật mã Hill

Tên mã	Mật mã Playfair	Mật mã Hill
Tổng quát	<p>-Mật mã Playfair sẽ thay thế từng cặp 2 ký tự trong bản rõ bởi 2 ký tự tương ứng trong ma trận khoá 5 x 5.</p> <p>-Lần lượt viết từng ký tự của khóa vào ma trận, từ trái sang phải, từ trên xuống dưới, bỏ các ký tự trùng lặp</p> <p>-Viết các ký tự còn lại trong bảng chữ cái vào ma trận theo thứ tự, I và J được coi như một ký tự</p> <p>-Mỗi ký tự trong cặp plaintext sẽ được mã hoá bằng ký tự nằm cùng hàng với nó, nhưng cùng cột với ký tự kia</p> <p>-Nếu cặp ký tự plaintext rơi vào cùng một hàng của ma trận thì mỗi ký tự được thay thế bởi ký tự bên phải nó.</p> <p>-Nếu ký tự plaintext rơi vào cột cuối cùng, thì ciphertext của nó là ký tự cùng hàng ở cột đầu tiên.</p>	<p>-Mật mã Hill sẽ thay thế từng nhóm <math>m</math> ký tự trong plaintext bởi <math>m</math> ký tự ciphertext</p> <p>-<math>m</math> ký tự ciphertext được xác định bởi hệ <math>m</math> phương trình tuyến tính</p> $C_m = (k_{m1}P_1 + k_{m2}P_2 + \dots + k_{mm}P_m) \bmod 26$

	<p>-Nếu cặp ký tự plaintext rơi vào chung một cột của ma trận thì mỗi ký tự được thay thế bởi ký tự ngay sát dưới.</p> <p>- Nếu ký tự plaintext rơi vào hàng cuối cùng, thì ciphertext của nó là ký tự cùng cột, ở hàng đầu tiên.</p> <p>-Nếu hai ký tự trong plaintext giống nhau thì chúng sẽ được cách ly bằng một ký tự đại diện, chẳng hạn là <b>x</b>.</p>	
Ưu	<p>Mật mã Playfair có không gian khoá lớn tương tự mật mã Monoalphabetic nên khó bị được khoá bằng phương pháp Brute -force</p> <p>Mật mã Playfair có khả năng che giấu một phần thông tin về tần suất xuất hiện các chữ cái, nhờ thực hiện mã hoá từng cặp hai ký tự</p>	<p>Độ an toàn của mật mã Hill sẽ càng lớn khi sử dụng ma trận K càng lớn</p> <p>Mật mã Hill có khả năng che dấu hoàn toàn tần suất xuất hiện các ký tự đơn</p> <p>Mật mã Hill rất mạnh khi chống lại tấn công chỉ biết ciphertext</p>
Nhược		<p>nhưng nó lại dễ dàng bị bẻ gãy với một tấn công biết plaintext, do có thể dễ dàng xác định ma trận K từ các cặp P-C đã biết.</p>

#### -Phân biệt Mật mã Vigenere, Vernman

Tên	Mật mã Vigenere	Vernman
Tổng quát	<p>Mật mã Vigenère sẽ thay thế từng nhóm <math>m</math> ký tự trong plaintext bởi <math>m</math> ký tự ciphertext</p> <p><math>m</math> ký tự ciphertext được xác định bởi hệ <math>m</math> phương trình</p> $C_m = (P_m + k_m) \bmod 26$ $P_m = (C_m - k_m) \bmod 26$	<p>Plaintext được biểu diễn dưới dạng một chuỗi bit nhị phân</p> <p>Khoá <math>K</math> cũng được biểu diễn dưới dạng một chuỗi bit nhị phân (càng dài càng tốt, càng ngẫu nhiên càng tốt)</p> <p>Ciphertext được sinh ra bởi phép XOR giữa plaintext với khoá <math>K</math></p>

Ưu	<p>Như vậy, nếu <math>k_1 = k_2 = \dots = k_m</math> thì mật mã Vigenère sẽ trở thành mật mã Caesar tổng quát.</p> <p>Khi <math>k_1 \neq k_2 \neq \dots \neq k_m</math>: một kí tự plaintext có thể được thay thế bởi nhiều kí tự khác nhau (ứng với các <math>k</math> khác nhau), nhờ vậy có thể che giấu được tần suất xuất hiện các kí tự.</p>	<p>Với khoá <math>K</math> đủ dài và ngẫu nhiên, các thông tin mang tính thống kê của ngôn ngữ có thể được che giấu hoàn toàn</p> <p>Sự ra đời của mật mã hệ nhị phân là tiền đề cho sự ra đời của các mật mã hiện đại.</p>
Nhược	<p>Mật mã Vigenère vẫn không giấu được hoàn toàn tần suất xuất hiện các kí tự, và vẫn có thể bị phân tích</p> <p>Giải pháp khắc phục là sử dụng hệ thống biểu diễn thông tin không mang tính thống kê của ngôn ngữ - đó là hệ nhị phân</p>	<p>Nếu <math>K</math> ngắn thì sẽ phải sử dụng <math>K</math> lặp đi lặp lại, làm giảm tính ngẫu nhiên, và có thể làm lộ một phần thông tin về thống kê tần suất.</p> <p>Tuy nhiên, việc sinh ra được một khoá <math>K</math> dài và thực sự ngẫu nhiên như vậy sẽ đòi hỏi nhiều công sức.</p>

#### -Phân biệt Virus với Worm

Virus	Worm
Virus là một đoạn mã chương trình có thể lây nhiễm tới các chương trình khác (bằng cách gắn bản sao của nó vào các chương trình mà nó tìm thấy)	Worm là một chương trình có thể tự nhân bản và gửi các bản sao của nó từ máy tính này đến máy tính khác qua các kết nối mạng.

#### -Phân biệt Virus Boot, Virus File thi hành, Virus Macro

Virus Boot	Virus File thi hành	Virus Macro
<p>Loại virus này lây lan vào đoạn mã khởi động trên Boot sector hay Master Boot Record của đĩa</p> <p>Khi máy tính khởi động, virus boot được kích hoạt.</p>	<p>Loại virus này lây nhiễm vào các file nhị phân thi hành được (.EXE, .COM, .DLL, .BIN, .SYS...)</p>	<p>Loại virus này lây nhiễm vào các macro trong các file tài liệu của MicroSoft Office (Word, Excel...)</p> <p>Một số macro có khả năng tự khi hành khi mở file, cất</p>

Nó sẽ thường trú trong bộ nhớ, chờ để lây vào một ổ đĩa mới Nếu đem ổ đĩa nhiễm virus lắp sang một máy tính khác, rồi khởi động máy từ ổ đĩa đó, máy tính sẽ bị nhiễm virus.	Đoạn mã virus có thể được gắn vào đầu file, cuối file, hoặc giữa file Khi file được chạy, virus sẽ được kích hoạt, nó sẽ tìm cách lây vào các file khác trong máy	file... Virus Macro thường nằm trong các macro tự động đó. Khi file được mở, virus sẽ được kích hoạt, sau đó nó tìm cách lây vào các file tài liệu khác.
---	--	---

#### -Worm

*Worm cũng có khả năng lây lan giống virus, nhưng nó khác với virus ở hai điểm cơ bản sau:*

+Worm là một chương trình hoàn chỉnh (chứ không phải là một đoạn mã gắn vào chương trình khác như virus)

+Kỹ thuật lây lan từ máy này sang máy khác của Worm dựa vào các lỗ hổng bảo mật của mạng máy tính.

1 số kỹ thuật lây lan Worm

+Tìm kiếm các hệ thống khác để lây nhiễm bằng cách khảo sát các bảng host hay những nơi chứa các địa chỉ các hệ thống từ xa.

+Thiết lập kết nối tới hệ thống từ xa.

+Tự sao chép đến hệ thống từ xa và kích hoạt để tự thi hành.

#### -Firewall

Firewall là một phương tiện hiệu quả để bảo vệ hệ thống khỏi các nguy cơ an ninh mạng, trong khi vẫn hỗ trợ các giao tiếp với bên ngoài qua các mạng diện rộng hay Internet.

Firewall được chèn vào giữa nội mạng và Internet để tạo nên một “bức tường chắn” điều khiển được, hình thành một vành đai bảo vệ.

#### -Phân biệt mật mã cổ điển và mật mã hiện đại

Tên	Mật mã cổ điển	Mật mã hiện đại
Thông tin	bao gồm kỹ thuật thay thế	bao gồm mật mã phi giao hoán và ẩn mã

	và kỹ thuật chuyển dịch hoán vị	<p>-Ẩn mã: Ẩn mã (Steganography) là một kỹ thuật giấu một văn bản, hình ảnh, file vào trong một văn bản, hình ảnh, file hay một vật phủ (là vật mang tin chứa ẩn mã) khác. Một trong những ưu điểm chính của ẩn mã là không thu hút sự chú ý, không bị sự “quan tâm” hay giám sát của đối phương. Điều này đặc biệt hữu ích cho các hoạt động tình báo thông tin hiện nay.</p> <p><b>Một số kỹ thuật ẩn mã:</b></p> <p>+<b>Phương pháp Vật lý</b></p> <p>+<b>Phương pháp sử dụng văn bản kỹ thuật số</b></p> <p>+<b>Sử dụng mạng xã hội</b></p> <p><b>Sử dụng mạng máy tính</b></p> <p>+<b>Sử dụng tài liệu đã được in ấn</b></p> <p>+<b>Sử dụng câu đố (puzzle)</b></p>

### Câu 11: SSL là gì

-SSL một tiêu chuẩn của công nghệ bảo mật, tạo ra một liên kết được mã hóa giữa máy chủ web và trình duyệt.

-Liên kết này đảm bảo tất cả các dữ liệu trao đổi giữa máy chủ web và trình duyệt luôn được bảo mật và an toàn.

-SSL được sử dụng rộng rãi trên hàng triệu Website ở khắp thế giới, giúp bảo vệ dữ liệu an toàn trên môi trường internet

-SSL cung cấp:

+Sự bảo mật: Các dữ liệu truyền tải giữa máy chủ và trình duyệt sẽ được mã hoá, đảm bảo tính riêng tư và toàn vẹn .

+ Khả năng chứng thực: Mỗi chứng chỉ số SSL được tạo ra cho một Website duy nhất, khẳng định độ tin cậy của Website đó.

+Một cơ quan uy tín sẽ xác thực danh tính và độ tin cậy của Website trước khi cấp chứng chỉ SSL cho Website.

### Câu 12: SET là gì

- SET (Secure Electronic Transaction – Giao dịch điện tử an toàn) được thiết kế để bảo vệ các giao dịch bằng thẻ tín dụng qua Internet
- SET là một tập các giao thức an ninh và các dạng thức cho phép người dùng thanh toán bằng thẻ tín dụng trong một mạng mở (như Internet) một cách an toàn.

-SET cung cấp 3 dịch vụ:

+Cung cấp kênh liên lạc an toàn giữa các thực thể góp mặt trong một giao dịch.

+Cung cấp tính tin cậy bằng cách dùng các chứng chỉ số X.509v3.

+Bảo đảm tính riêng tư, do các thông tin chỉ được cung cấp cho các thực thể khi tham gia giao dịch và vào lúc cần thiết.

-Các đặc điểm chính của SET

+Tính tin cậy của thông tin

Các thông tin về người dùng thẻ và tài khoản tương ứng được bảo vệ an toàn khi lưu thông trên mạng

Ngay cả người bán cũng không thể nắm giữ được số thẻ tín dụng của người sở hữu, số hiệu này chỉ được cung cấp cho ngân hàng phát hành thẻ

Mã hóa DES được sử dụng để cung cấp tính tin cậy

+Tính toàn vẹn dữ liệu

Thông tin thanh toán được gửi từ một người dùng thẻ tới nhà bán hàng bao gồm các thông tin đặt hàng, dữ liệu cá nhân, và các lệnh chi trả.

SET đảm bảo rằng nội dung các thông điệp này không bị biến đổi trên đường đi.

Chữ ký số RSA dùng mã hash SHA-1 được sử dụng để đảm bảo tính toàn vẹn của thông điệp. Ngoài ra có thể bảo vệ các thông điệp khác bằng HMAC dùng SHA-1

+Chứng thực tài khoản của người dùng thẻ

SET cho phép các nhà bán hàng xác minh một người dùng thẻ là hợp pháp đối với một thẻ tín dụng hợp lệ và tài khoản ứng với thẻ đó.

Chứng chỉ số X.509v3 chứa các chữ ký RSA được sử dụng cho chức năng này.

+Chứng thực nhà bán hàng

SET cho phép người sở hữu thẻ xác minh được rằng nhà bán hàng đã sẵn có mối quan hệ với cơ quan tài chính hỗ trợ thanh toán bằng thẻ.

SET dùng các chứng chỉ số X.509v3 với các chữ ký RSA cho chức năng này



**Câu 13: Khi mua hàng người bán gửi cho người mua thông tin:**

+Khách hàng chọn các món hàng cần mua rồi gửi danh sách hàng hoá cho người bán. Người bán gửi lại khách một đơn hàng có chứa danh mục hàng hóa đã chọn, số lượng, đơn giá và tổng số tiền.

+Người bán gửi một bản sao chứng chỉ , nhờ vậy khách hàng có thể xác minh được rằng mình đang giao dịch với một cửa hàng có thực và hợp lệ.

+ Người bán gửi xác nhận đơn hàng cho khách hàng.

+ Người bán cung cấp hàng hóa hoặc dịch vụ. Người bán chuyên chở hàng hóa hoặc áp dụng dịch vụ cho khách hàng.

**Câu 14: Chữ ký kép:**

+Khách hàng muốn gửi thông tin đơn hàng (Order Information - OI) cho người bán và thông tin thanh toán (Payment Information - PI) cho ngân hàng.

+Người bán không cần phải biết mã thẻ thanh toán của khách hàng, và ngân hàng không cần phải biết chi tiết đơn hàng của khách hàng.

+ Chữ ký kép nhằm liên kết OI và PI của một giao dịch, tránh ghép nhầm OI của giao dịch này với PI của giao dịch khác, làm cơ sở để giải quyết tranh chấp sau này.

**Câu 15: Dịch vụ chứng thực X.509**

-X.509 là một tiêu chuẩn về chứng chỉ khoá công khai được chấp nhận rộng rãi trên toàn thế giới

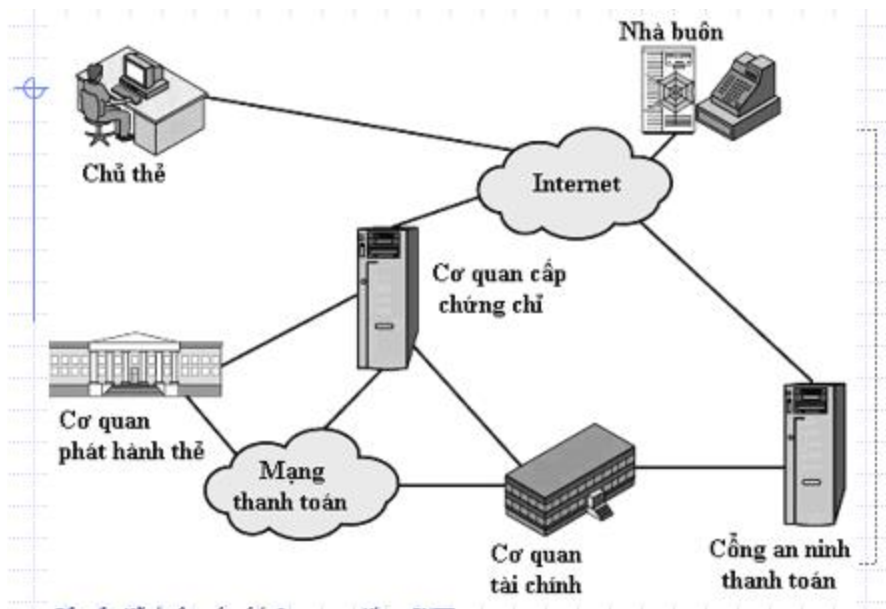
-Các chứng chỉ X.509 được sử dụng trong hầu hết các ứng dụng về an ninh mạng

-Mỗi chứng chỉ X.509 ứng với một người dùng, được tạo ra bởi CA (Tổ chức cấp chứng chỉ tin cậy)

Các chứng chỉ sẽ được đặt vào một thư mục công khai của CA, hoặc đặt vào thư mục của người dùng

Người dùng có thể truy xuất thư mục của CA để lấy chứng chỉ của người khác, hay tự phân phối chứng chỉ của mình cho người khác, mà không lo bị giả mạo.

**Câu 15 Các thành phần tham gia giao dịch điện tử**



Câu 16: Các bước thực hiện giao dịch

- 1. Khách hàng mở tài khoản.** Khách hàng mở một tài khoản thẻ tín dụng, tại một ngân hàng có hỗ trợ thanh toán điện tử và SET.
- 2. Khách hàng nhận được một chứng chỉ.** Sau khi xác minh nhận dạng thích hợp, khách hàng nhận được một chứng chỉ số X.509v3, được ngân hàng ký. Chứng chỉ này thiết lập một mối quan hệ giữa cặp khóa của khách hàng với thẻ tín dụng được cấp phát.
- 3. Nhà bán hàng có các chứng chỉ của mình.** Một người bán hàng phải sở hữu hai chứng chỉ cho hai cặp khóa-công-khai của mình: một để ký các thông điệp và một để trao đổi khóa.
- 4. Khách hàng đặt đơn hàng.** Khách hàng chọn các món hàng cần mua rồi gửi danh sách hàng hoá cho người bán. Người bán gửi lại khách một đơn hàng có chứa danh mục hàng hoá đã chọn, số lượng, đơn giá và tổng số tiền.
- 5. Người bán được xác minh.** Cùng với đơn hàng, người bán hàng gửi một bản sao chứng chỉ, nhờ vậy khách hàng có thể xác minh được rằng mình đang giao dịch với một cửa hàng có thực và hợp lệ.
- 6. Gửi đơn hàng và thông tin thanh toán.** Khách hàng gửi cả đơn hàng và các thông tin thanh toán cho người bán, cùng với chứng chỉ khách hàng của mình. Thông tin thanh toán (bao gồm chi tiết về thẻ tín dụng) được mã hóa sao cho người bán hàng không thể đọc được. Chứng chỉ khách hàng cho phép người bán xác minh khách hàng.

**7. Người bán yêu cầu cấp phép thanh toán.** Người bán gửi thông tin thanh toán cho công an ninh thanh toán, yêu cầu xác nhận chủ thẻ có khả năng thanh toán đơn hàng.

**8. Người bán xác nhận đơn hàng.** Người bán gửi xác nhận đơn hàng cho khách hàng.

**9. Người bán cung cấp hàng hóa hoặc dịch vụ.** Người bán chuyên chở hàng hóa hoặc áp dụng dịch vụ cho khách hàng.

**10. Người bán yêu cầu thanh toán.** Yêu cầu này được gửi tới công an ninh thanh toán là nơi giải quyết tất cả các thủ tục thanh toán

### Câu 17: Các kỹ thuật lấy mật khẩu

1. Thử các mật khẩu từng dùng với các tài khoản chuẩn gắn liền với hệ thống. Rất nhiều quản trị viên đã không bận tâm đến việc thay đổi các mật khẩu này.
2. Thử hết tất cả các mật khẩu ngắn (một đến ba ký tự).
3. Thử các từ trong từ điển trực tuyến của hệ thống, hoặc một danh sách mà rất có khả năng được sử dụng làm các mật khẩu.
4. Thu thập thông tin người dùng, chẳng hạn tên đầy đủ, tên vợ hoặc chồng cùng con cái, các bức tranh có trong văn phòng của họ, những cuốn sách ở văn phòng liên quan tới sở thích riêng...
5. Thử số điện thoại của người dùng, các số An sinh xã hội và số phòng ở, phòng làm việc.
6. Thử tất cả các biển đăng ký xe hợp pháp của địa phương.
7. Dùng một Trojan horse để ăn cắp mật khẩu.
8. Đấu nối một đường truyền giữa người dùng từ xa với host hệ thống.

### Câu 18: $A \rightarrow B: E(K, M) || H(M) || E(PUB, K)$

- Bên gửi :
  - A gửi B đoạn mã có : Thông điệp đầu vào M được mã hóa bởi hàm E với khóa K thu được  $E(K, M)$ , sau đó lại được ghép với khóa K được mã hóa bởi hàm E với khóa công khai PUB
- Bên nhận:
  - B nhận được đoạn mã từ A :

+Đưa hàm mã hóa E qua hàm giải mã với khóa riêng PRB thu được khóa K. Rồi sau đó đem khóa K giải mã hàm E(K, M) thu được thông điệp M. Sau đó đưa M qua hàm hash thu được H(M)(1)

- So sánh 2 hàm Hash (1)&(2) để xác thực nội dung thông điệp có bị thay đổi trên đường truyền hay ko.
- Tác dụng:
  - Xác thực nội dung thông điệp có chính xác hay ko
  - Có tác dụng giữ bí mật(PUB))

### Câu 19 A->B: E(K,M)||H(E(K, M))

- Bên gửi :
  - A gửi B đoạn mã có : Thông điệp đầu vào M được mã hóa bởi hàm E với khóa K thu được E(K, M), sau đó ghép với thông điệp M được mã hóa bởi hàm E với khóa K, rồi sau đó toàn bộ hàm E(K, M) đưa qua hàm hash H thu được mã H(E(K, M))
- Bên nhận:
  - B nhận được đoạn mã từ A :
    - +Đưa hàm mã hóa E(K, M) đưa qua hàm hash thu được mã hash H(E(K, M)) (1)
    - So sánh 2 thông điệp Hash (1)&(2) để xác thực nội dung thông điệp có bị thay đổi trên đường truyền hay ko
- Tác dụng:
  - Xác thực nội dung thông điệp có chính xác không

### Câu 20 A->B: E(K2, [M||C(K1, M)])||E(PUB, [K1||K2])

- Bên gửi :
  - A gửi B đoạn mã có : Thông điệp M được ghép mã MAC bởi hàm C với mật khóa K1 thu được C(K1, M) sau đó được ghép với thông điệp M và toàn bộ đoạn mã được mã hóa bởi hàm E với khóa K2. Sau đó toàn bộ đoạn mã ghép với hàm E gồm Khóa K2 ghép với khóa K1 được mã hóa bởi khóa công khai PUB
- Bên nhận:
  - B nhận được đoạn mã từ A :
    - Tách ra làm 2 phần:
    - +Giải mã hàm E bằng khóa riêng PRB thu được 2 khóa K1 và K2
    - + Giải mã hàm E biết khóa K2 thu được thông điệp M và hàm C(K1, M)(2)

+ Mã hóa hàm MAC với khóa K1 và thông điệp M thu được hàm  $C(K1, M)(1)$

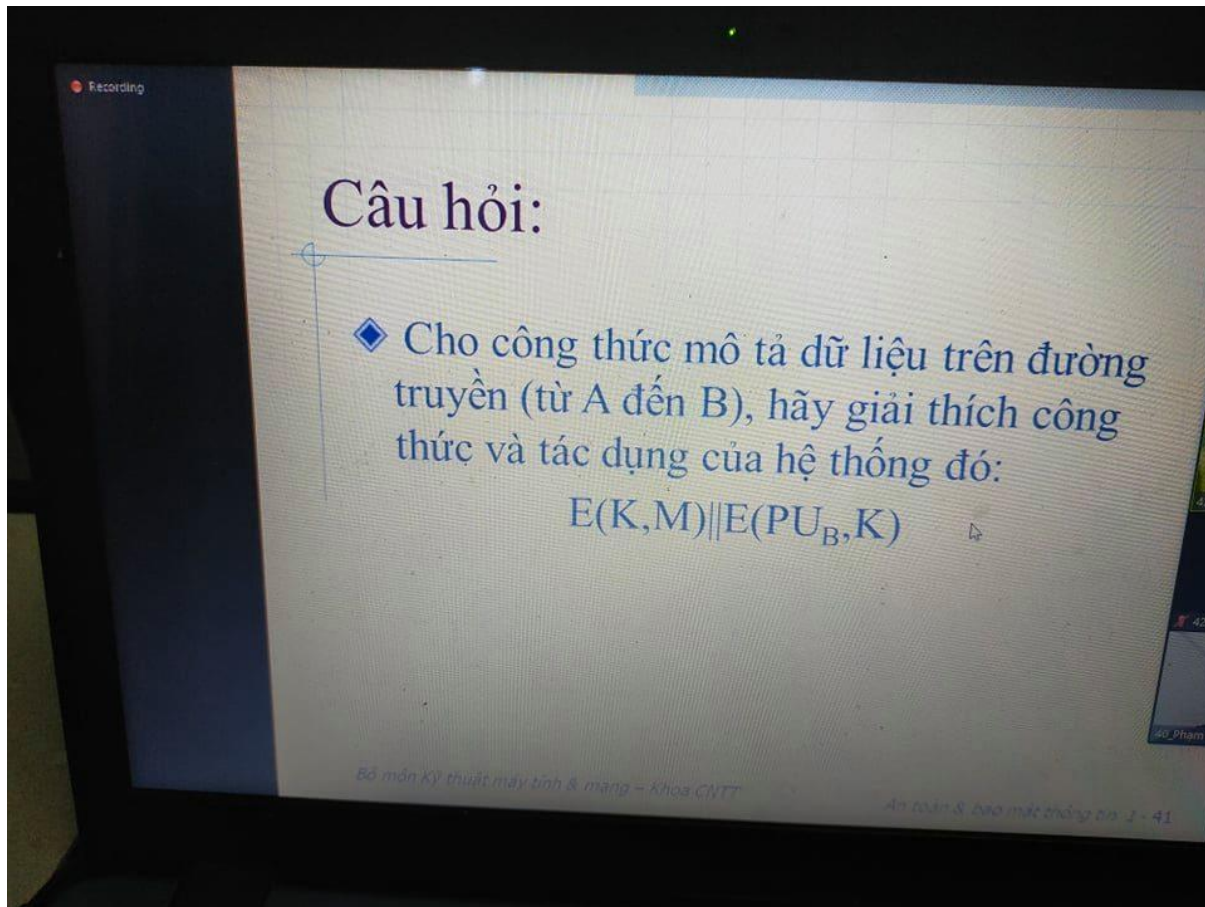
So sánh 2 hàm MAC (1)&(2) để xác thực nội dung thông điệp có bị thay đổi trên đường truyền hay ko

- Tác dụng:
  - Xác thực nội dung thông điệp có chính xác không
  - Có tác dụng giữ bí mật

Câu 21 A->B:  $E(K1, M) || C(K2, E(K1, M)) || E(PUB, [K1 || K2])$

- Bên gửi :
  - A gửi B đoạn mã có : Thông điệp M được mã hóa bởi hàm E với khóa K1 sau đó đoạn mã ghép với Thông điệp M được mã hóa bởi hàm E với mật khóa K1 sau đó hàm E đưa qua hàm MAC với mật khóa K2 thu được  $C(K2, E(K1, M))$ . Sau đó toàn bộ đoạn mã ghép với hàm E gồm Khóa K2 ghép với khóa K1 được mã hóa bởi khóa công khai PUB
- Bên nhận:
  - B nhận được đoạn mã từ A :
  - Tách ra làm 3 phần
    - +Giải mã hàm E bằng khóa riêng PRB thu được 2 khóa K1 và K2
    - + Giải mã hàm E biết khóa K1 thu được thông điệp M
    - + Đưa hàm  $E(K1, M)$  qua hàm MAC với khóa K2 thu được hàm  $MAC(C(K2, E(K1, M)))(1)$
  - So sánh 2 hàm MAC (1)&(2) để xác thực nội dung thông điệp có bị thay đổi trên đường truyền hay ko
- Tác dụng:
  - Xác thực nội dung thông điệp có chính xác không
  - Có tác dụng giữ bí mật

# Đề



- Bên gửi :
  - A gửi B đoạn mã có : Thông điệp đầu vào M được mã hóa bởi hàm E với khóa K thu được E(K, M), sau đó lại được ghép với khóa K được mã hóa bởi hàm E với khóa công khai PUB
  -

- Bên nhận:
    - B dùng khóa riêng PRb để giải mã thu được khóa K
    - Dùng khóa K để giải mã hàm E đầu tiên thu được thông điệp M từ đó B biết đc NỘI DUNG thông điệp M
- =>Tác dụng
- Bảo mật khóa K nhờ có khóa công khai PUB

Bảo mật thông tin nhờ có Pub

Virus	Worm
-------	------

<p>Virus là một đoạn mã chương trình có thể lây nhiễm tới các chương trình khác (bằng cách gắn bản sao của nó vào các chương trình mà nó tìm thấy)</p>	<p>Worm là một chương trình có thể tự nhân bản và gửi các bản sao của nó từ máy tính này đến máy tính khác qua các kết nối mạng.</p>
--	--